FIG.1

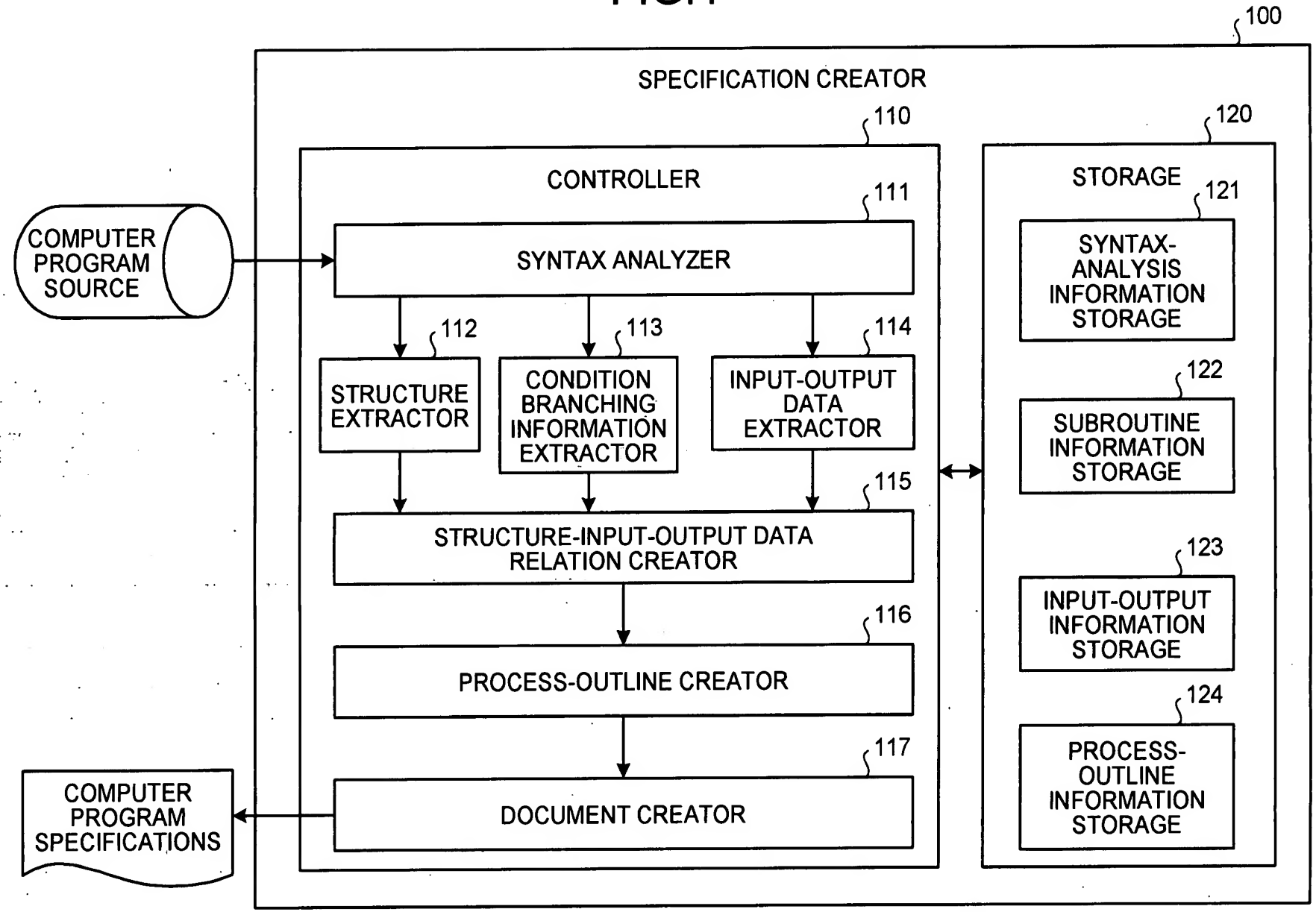

100
SPECIFICATION CREATOR
110
CONTROLLER
111
SYNTAX ANALYZER
112
STRUCTURE EXTRACTOR
113
CONDITION BRANCHING INFORMATION EXTRACTOR
114
INPUT-OUTPUT DATA EXTRACTOR
115
STRUCTURE-INPUT-OUTPUT DATA RELATION CREATOR
116
PROCESS-OUTLINE CREATOR
117
DOCUMENT CREATOR
120
STORAGE
121
SYNTAX-ANALYSIS INFORMATION STORAGE
122
SUBROUTINE INFORMATION STORAGE
123
INPUT-OUTPUT INFORMATION STORAGE
124
PROCESS-OUTLINE INFORMATION STORAGE
COMPUTER PROGRAM SOURCE
COMPUTER PROGRAM SPECIFICATIONS

FIG.2

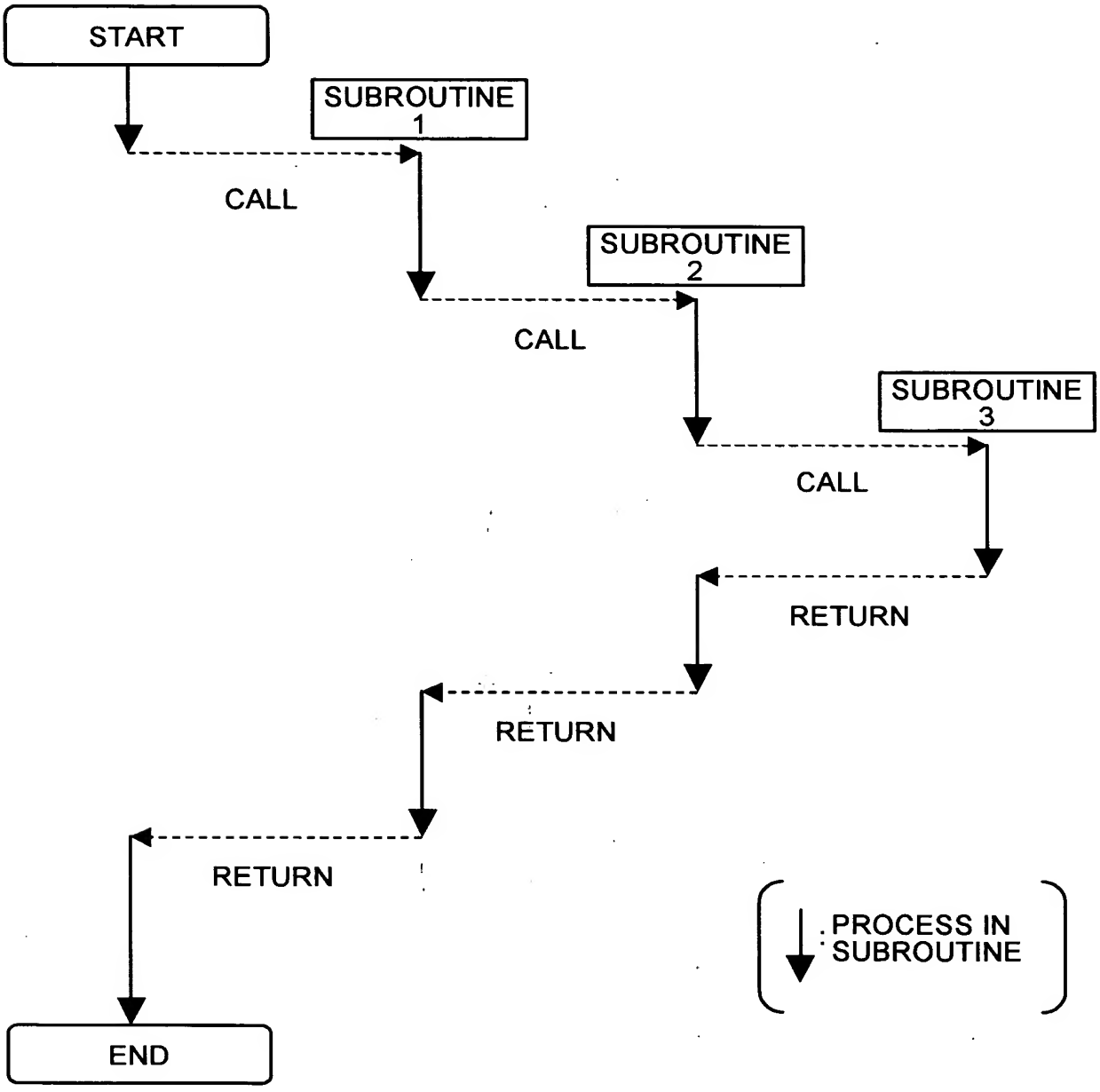

START
SUBROUTINE 1
CALL
SUBROUTINE 2
CALL
SUBROUTINE 3
CALL
RETURN
RETURN
RETURN
END
PROCESS IN SUBROUTINE

FIG.3

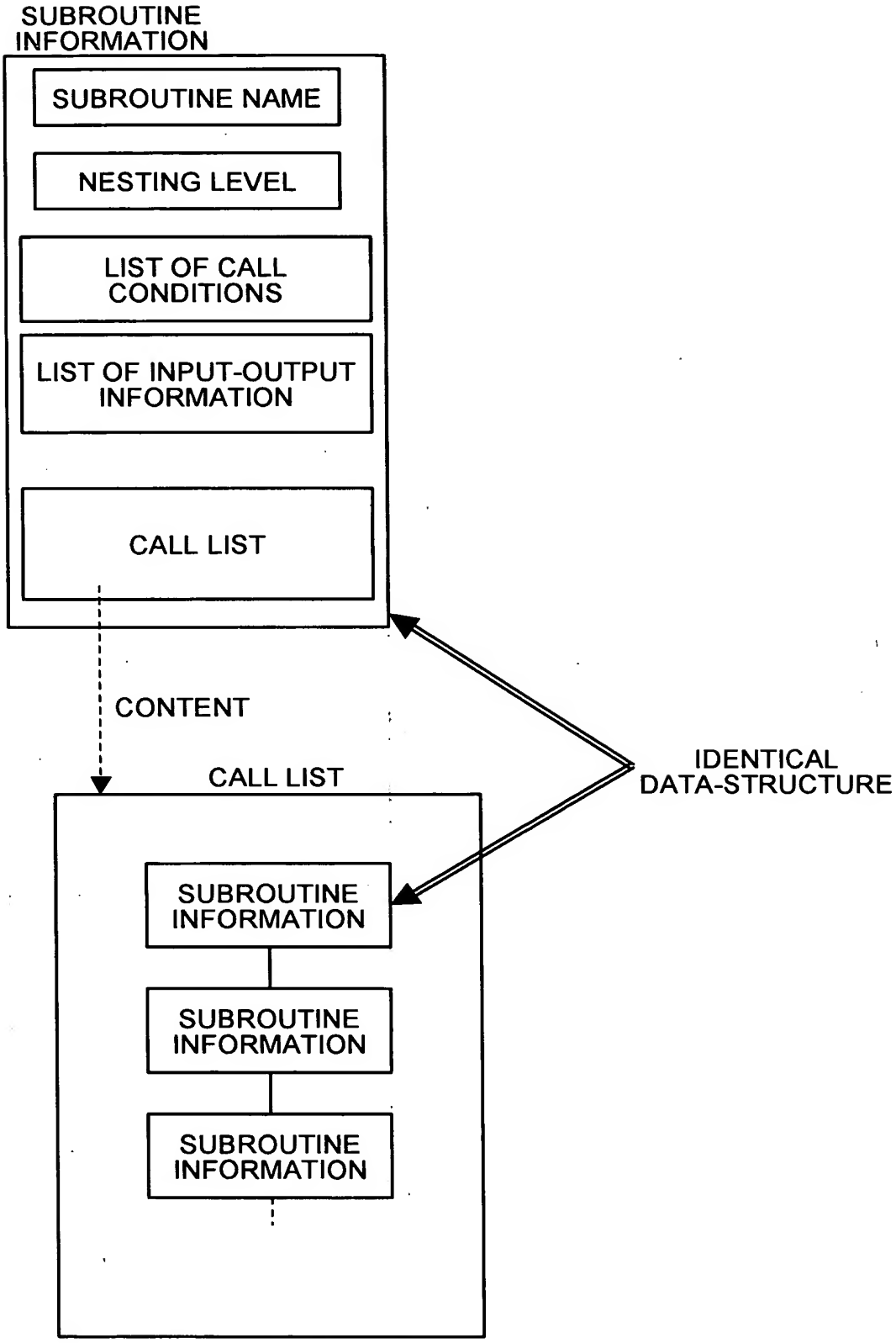

SUBROUTINE INFORMATION
SUBROUTINE NAME
NESTING LEVEL
LIST OF CALL CONDITIONS
LIST OF INPUT-OUTPUT INFORMATION
CALL LIST
CONTENT
CALL LIST
IDENTICAL DATA-STRUCTURE
SUBROUTINE INFORMATION
SUBROUTINE INFORMATION
SUBROUTINE INFORMATION

FIG.4

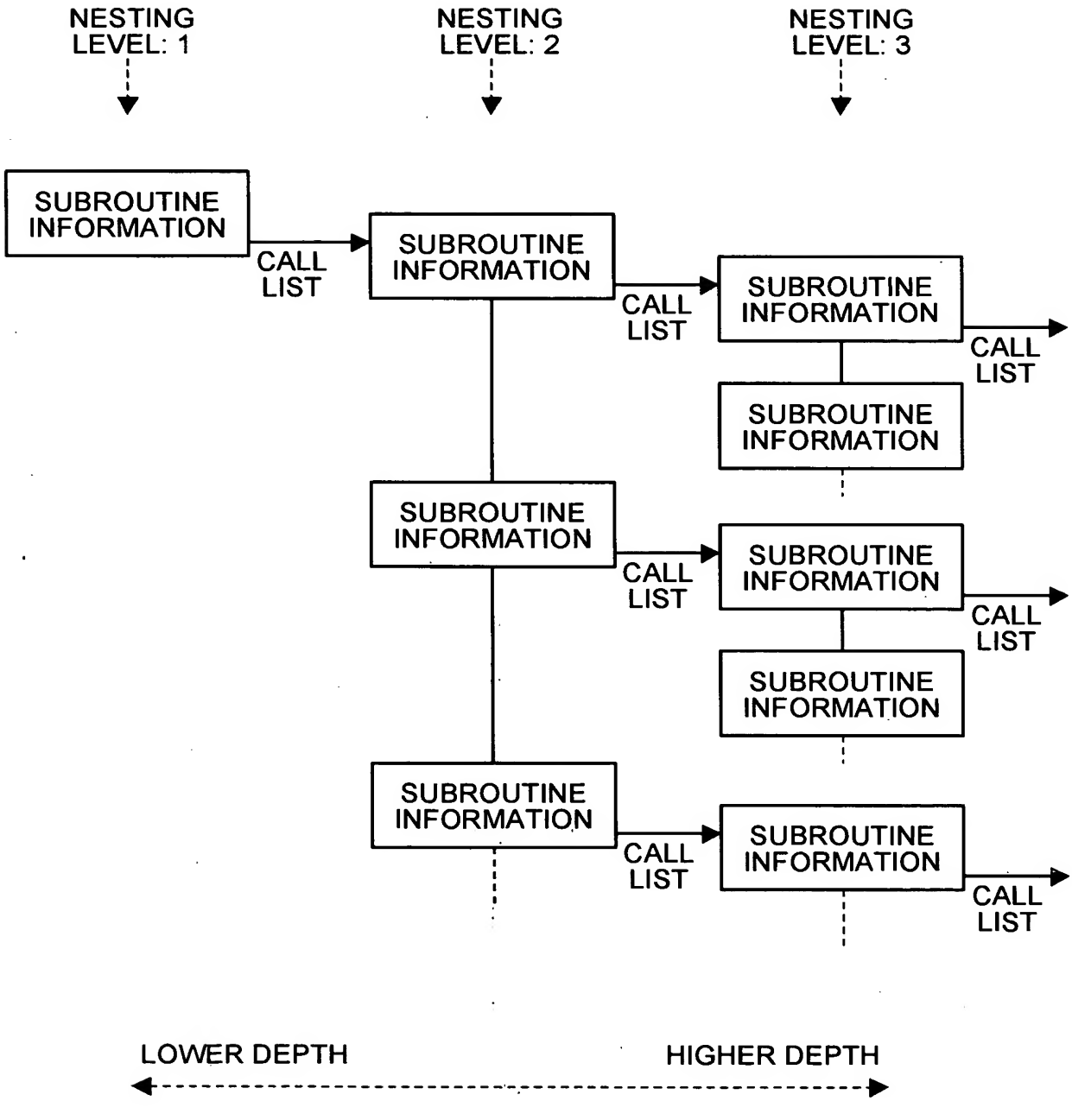

NESTING LEVEL: 1
NESTING LEVEL: 2
NESTING LEVEL: 3
SUBROUTINE INFORMATION
CALL LIST
SUBROUTINE INFORMATION
CALL LIST
SUBROUTINE INFORMATION
CALL LIST
SUBROUTINE INFORMATION
SUBROUTINE INFORMATION
CALL LIST
SUBROUTINE INFORMATION
CALL LIST
SUBROUTINE INFORMATION
SUBROUTINE INFORMATION
CALL LIST
SUBROUTINE INFORMATION
CALL LIST
LOWER DEPTH
HIGHER DEPTH

FIG.5

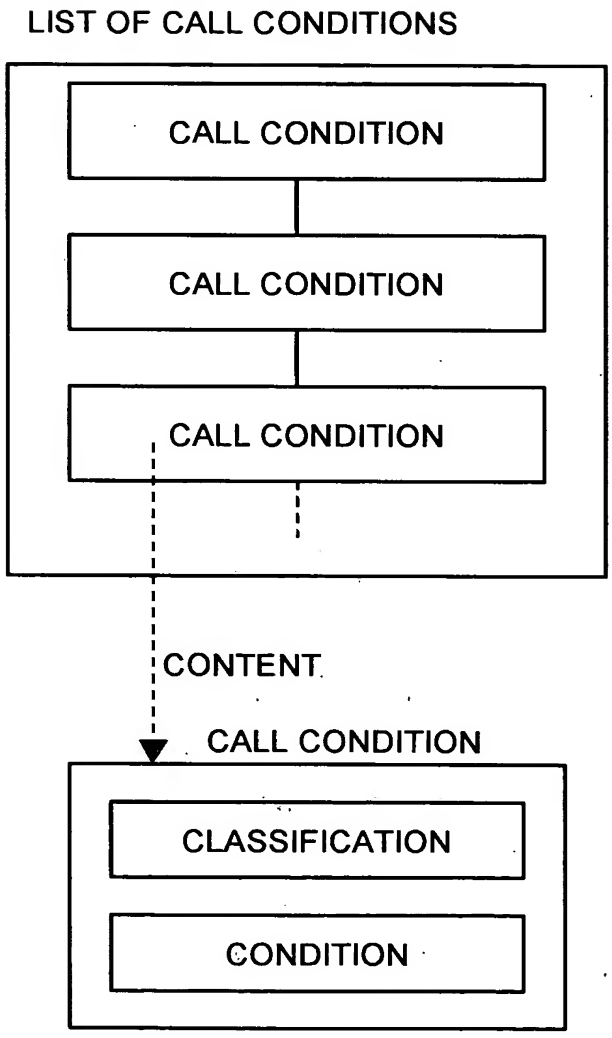

LIST OF CALL CONDITIONS
CALL CONDITION
CALL CONDITION
CALL CONDITION
CONTENT
CALL CONDITION
CLASSIFICATION
CONDITION

FIG.6

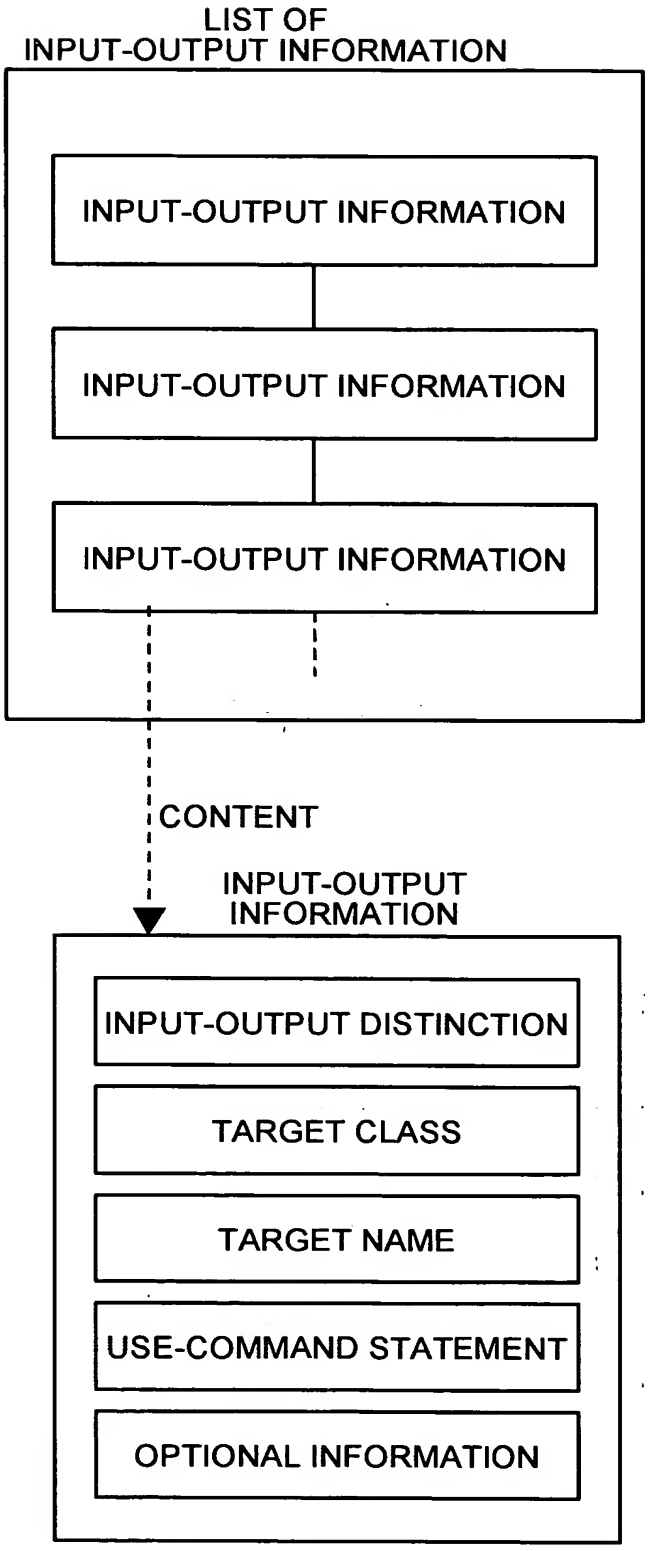

LIST OF
INPUT-OUTPUT INFORMATION
INPUT-OUTPUT INFORMATION
INPUT-OUTPUT INFORMATION
INPUT-OUTPUT INFORMATION
CONTENT
INPUT-OUTPUT
INFORMATION
INPUT-OUTPUT DISTINCTION
TARGET CLASS
TARGET NAME
USE-COMMAND STATEMENT
OPTIONAL INFORMATION

FIG.7

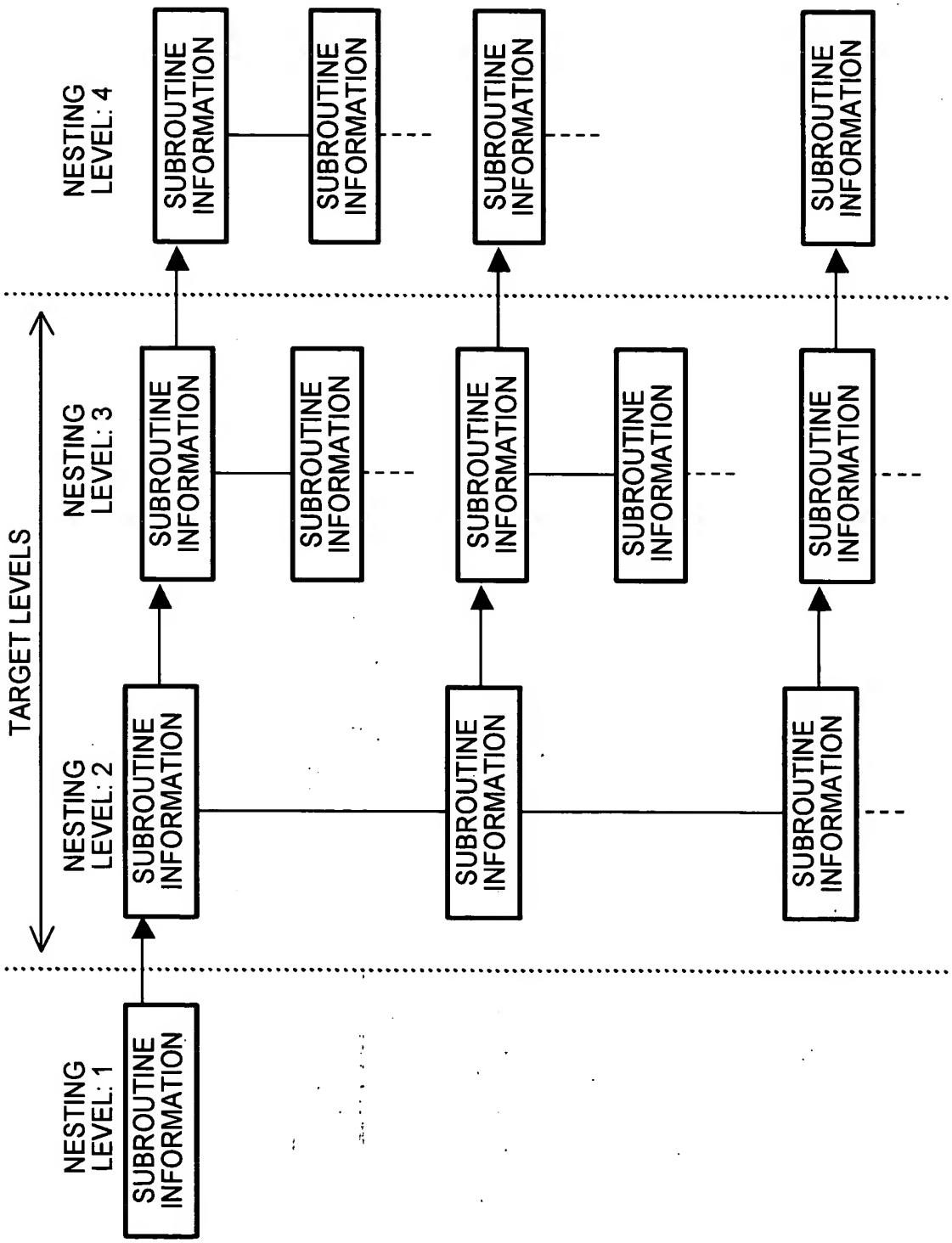

TARGET LEVELS
NESTING LEVEL: 1
NESTING LEVEL: 2
NESTING LEVEL: 3
NESTING LEVEL: 4
SUBROUTINE INFORMATION

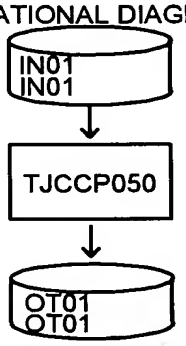
FIG.8

COMPUTER PROGRAM NAME	TJCCP050	FILE NAME	TJCCP050.scob

[COLUMN FOR COMMENTS]

ENTER BUSINESS CONTENTS ETC.

[INPUT-OUTPUT RELATIONAL DIAGRAM]



[COMMON AREA]

No.	RECORD NAME	DETAILS
1	SQLSTATE	
2	SQLMSG	

[FILE INFORMATION]

No.	FILE NAME	EXTERNAL UNIT NAME	CLASS	ORGANI-ZATION	LIBRARY NAME	DETAILS
1	IN01	IN01	COBOL FILE	ROW ORDER	TJCCF031.cbl	
2	OT01	OT01	COBOL FILE	ROW ORDER	TJCCF051.cbl	

[INPUT-OUTPUT AND SECTION NAME]

[INPUT]
TJSC. REQUEST FILE, TJSC. MOVE-IN FILE, TJSC. UNIT MASTER, TJSC. RENT CLASSIFICATION MASTER, TJSC. SETTLED VALUE FIXING FILE, TJSC. NAME FILE, TJSC. CODE MASTER, SC_B101001. MB101012, SC_B101001. MB101014, INO1
[PROCESS]
PARAMETERS-CHECKING PROCESS, READING PROCESS, DETAIL PROCESS (DATA-CHECKING PROCESS, SETTLED VALUE (FIXING) F READING PROCESS, NAME F READING PROCESS, AND EDITING PROCESS)
[OUTPUT]
TJCCF051

[PROCESS STRUCTURE DIAGRAM]
(DISPLAY FROM SECOND LAYER TO THIRD LAYER OF SECTION)

←CALLED BY	SECTION CALL STRUCTURE]	CALLED FROM →
PARAMETERS-CHECKING PROCESS		
SALES PROCESS		
DETAILED PROCESS	DATA-CHECKING PROCESS	

FIG.9

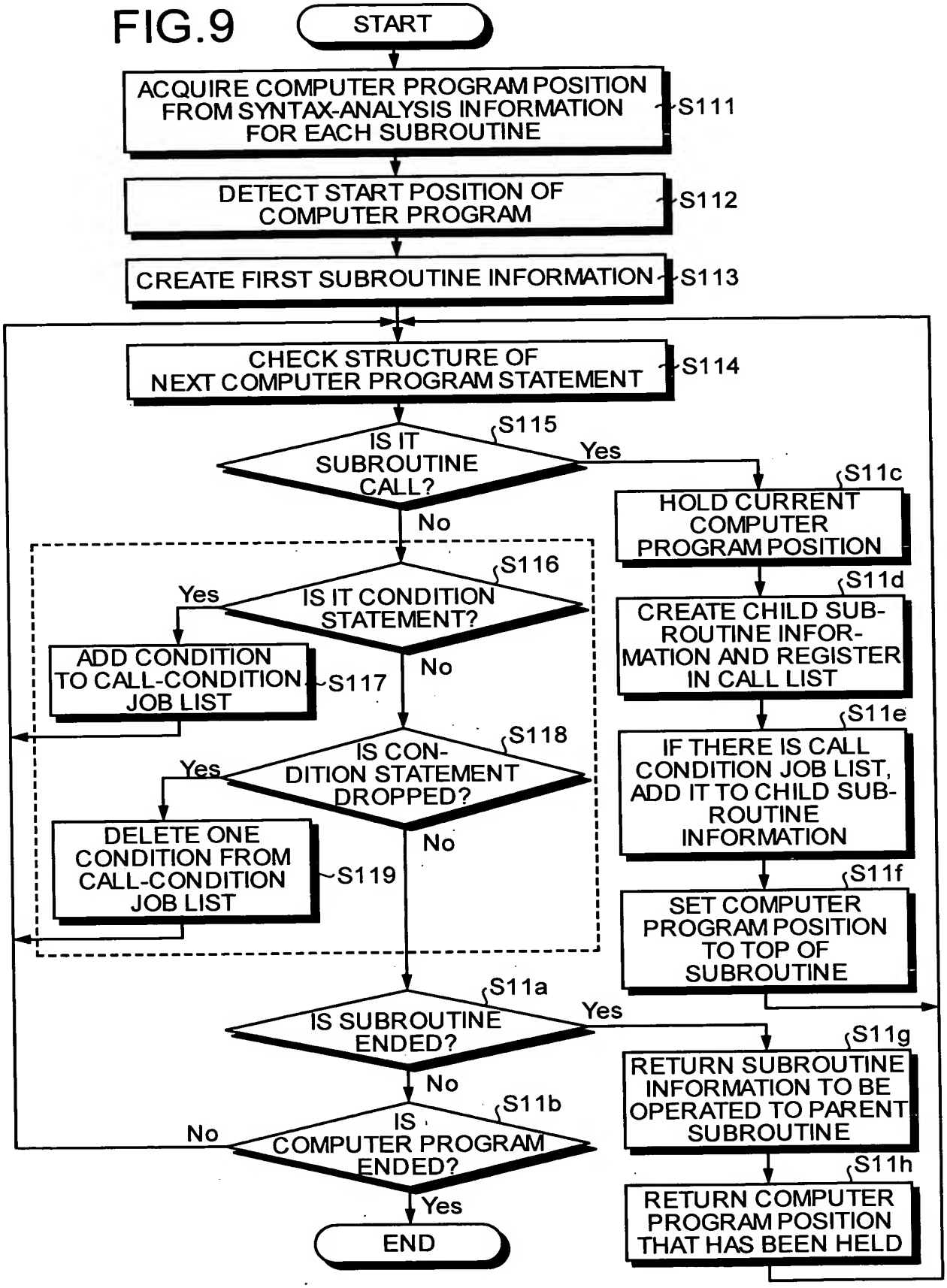

START
ACQUIRE COMPUTER PROGRAM POSITION FROM SYNTAX-ANALYSIS INFORMATION FOR EACH SUBROUTINE
S111
DETECT START POSITION OF COMPUTER PROGRAM
S112
CREATE FIRST SUBROUTINE INFORMATION
S113
CHECK STRUCTURE OF NEXT COMPUTER PROGRAM STATEMENT
S114
IS IT SUBROUTINE CALL?
S115
Yes
No
HOLD CURRENT COMPUTER PROGRAM POSITION
S11c
CREATE CHILD SUB-ROUTINE INFOR-MATION AND REGISTER IN CALL LIST
S11d
IF THERE IS CALL CONDITION JOB LIST, ADD IT TO CHILD SUB-ROUTINE INFORMATION
S11e
SET COMPUTER PROGRAM POSITION TO TOP OF SUBROUTINE
S11f
IS IT CONDITION STATEMENT?
S116
Yes
No
ADD CONDITION TO CALL-CONDITION JOB LIST
S117
IS CON-DITION STATEMENT DROPPED?
S118
Yes
No
DELETE ONE CONDITION FROM CALL-CONDITION JOB LIST
S119
IS SUBROUTINE ENDED?
S11a
Yes
No
RETURN SUBROUTINE INFORMATION TO BE OPERATED TO PARENT SUBROUTINE
S11g
IS COMPUTER PROGRAM ENDED?
S11b
No
Yes
RETURN COMPUTER PROGRAM POSITION THAT HAS BEEN HELD
S11h
END

FIG.10

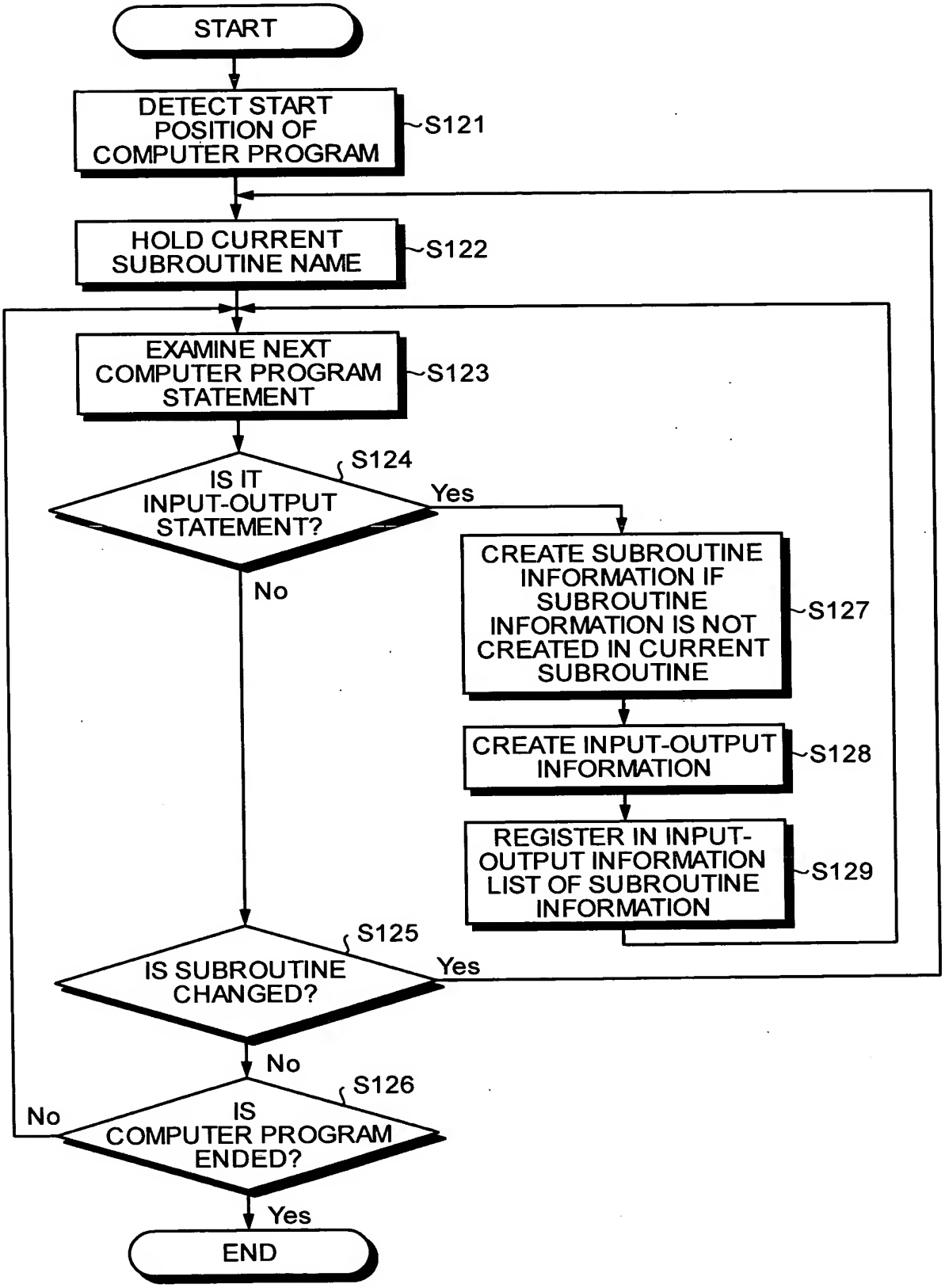

START
DETECT START POSITION OF COMPUTER PROGRAM
S121
HOLD CURRENT SUBROUTINE NAME
S122
EXAMINE NEXT COMPUTER PROGRAM STATEMENT
S123
IS IT INPUT-OUTPUT STATEMENT?
S124
Yes
No
CREATE SUBROUTINE INFORMATION IF SUBROUTINE INFORMATION IS NOT CREATED IN CURRENT SUBROUTINE
S127
CREATE INPUT-OUTPUT INFORMATION
S128
REGISTER IN INPUT-OUTPUT INFORMATION LIST OF SUBROUTINE INFORMATION
S129
IS SUBROUTINE CHANGED?
S125
Yes
No
IS COMPUTER PROGRAM ENDED?
S126
No
Yes
END

FIG.11

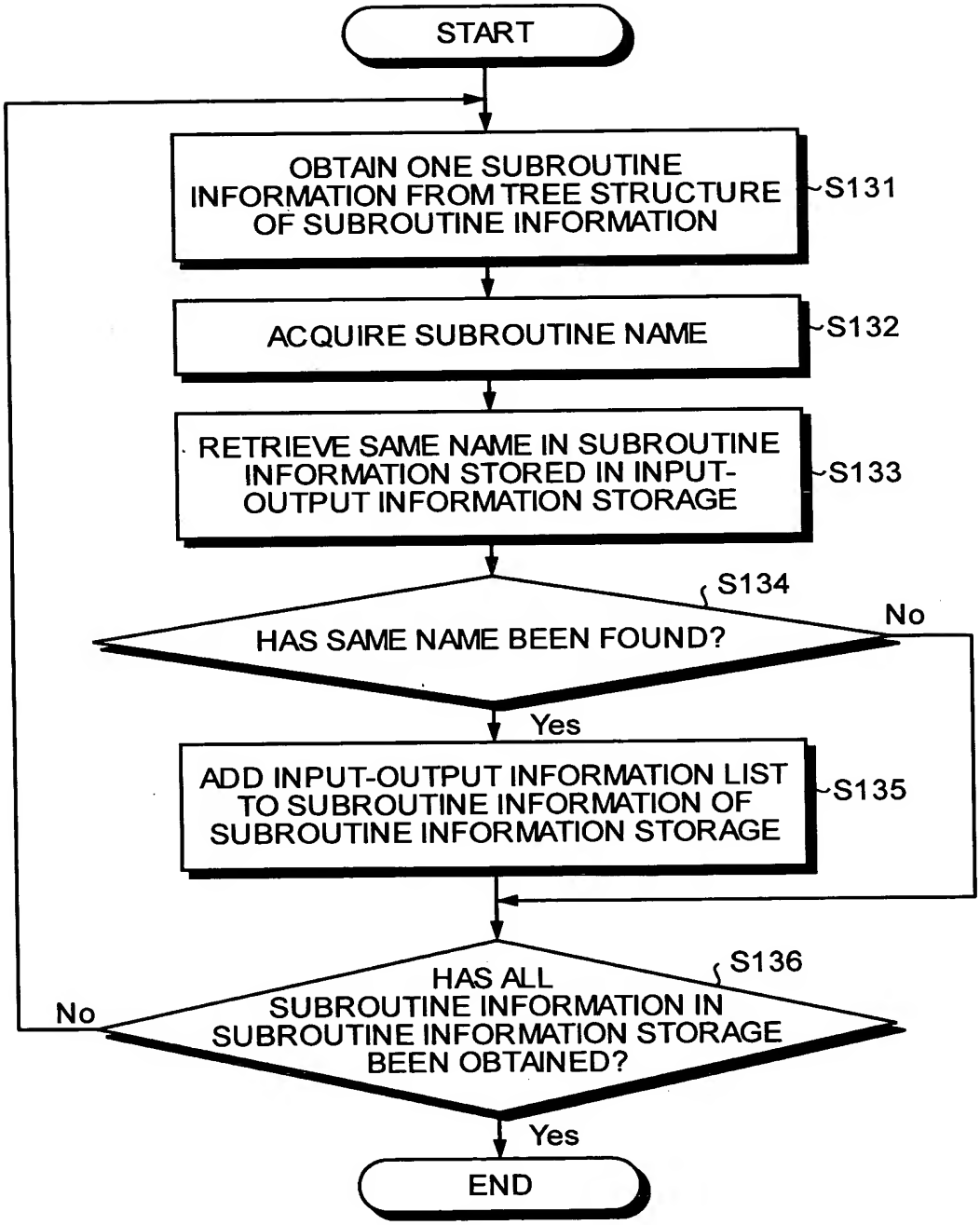

START
OBTAIN ONE SUBROUTINE INFORMATION FROM TREE STRUCTURE OF SUBROUTINE INFORMATION
S131
ACQUIRE SUBROUTINE NAME
S132
RETRIEVE SAME NAME IN SUBROUTINE INFORMATION STORED IN INPUT-OUTPUT INFORMATION STORAGE
S133
S134
HAS SAME NAME BEEN FOUND?
No
Yes
ADD INPUT-OUTPUT INFORMATION LIST TO SUBROUTINE INFORMATION OF SUBROUTINE INFORMATION STORAGE
S135
S136
HAS ALL SUBROUTINE INFORMATION IN SUBROUTINE INFORMATION STORAGE BEEN OBTAINED?
No
Yes
END

FIG.12

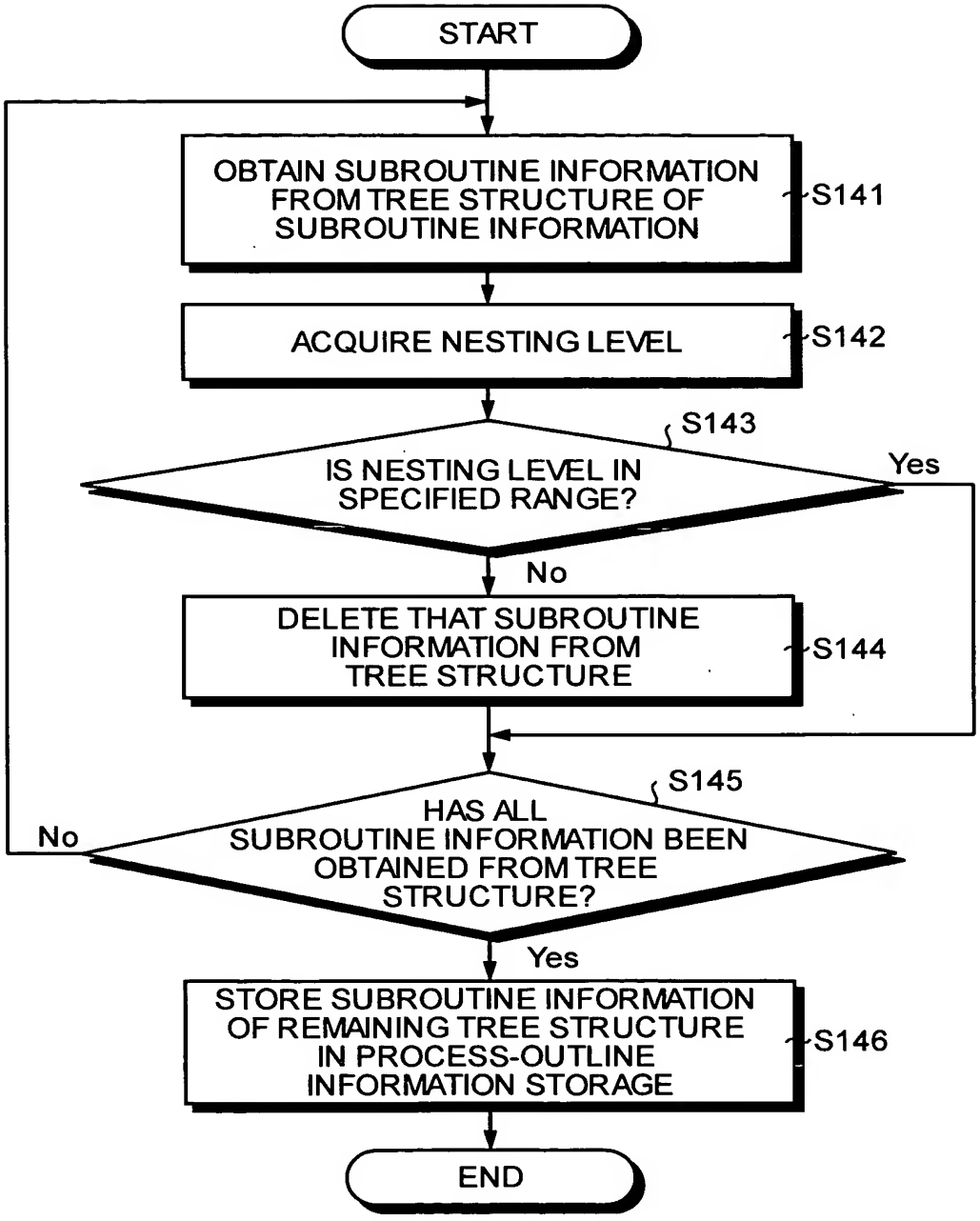

START
OBTAIN SUBROUTINE INFORMATION FROM TREE STRUCTURE OF SUBROUTINE INFORMATION
S141
ACQUIRE NESTING LEVEL
S142
S143
IS NESTING LEVEL IN SPECIFIED RANGE?
Yes
No
DELETE THAT SUBROUTINE INFORMATION FROM TREE STRUCTURE
S144
S145
HAS ALL SUBROUTINE INFORMATION BEEN OBTAINED FROM TREE STRUCTURE?
No
Yes
STORE SUBROUTINE INFORMATION OF REMAINING TREE STRUCTURE IN PROCESS-OUTLINE INFORMATION STORAGE
S146
END

FIG.13

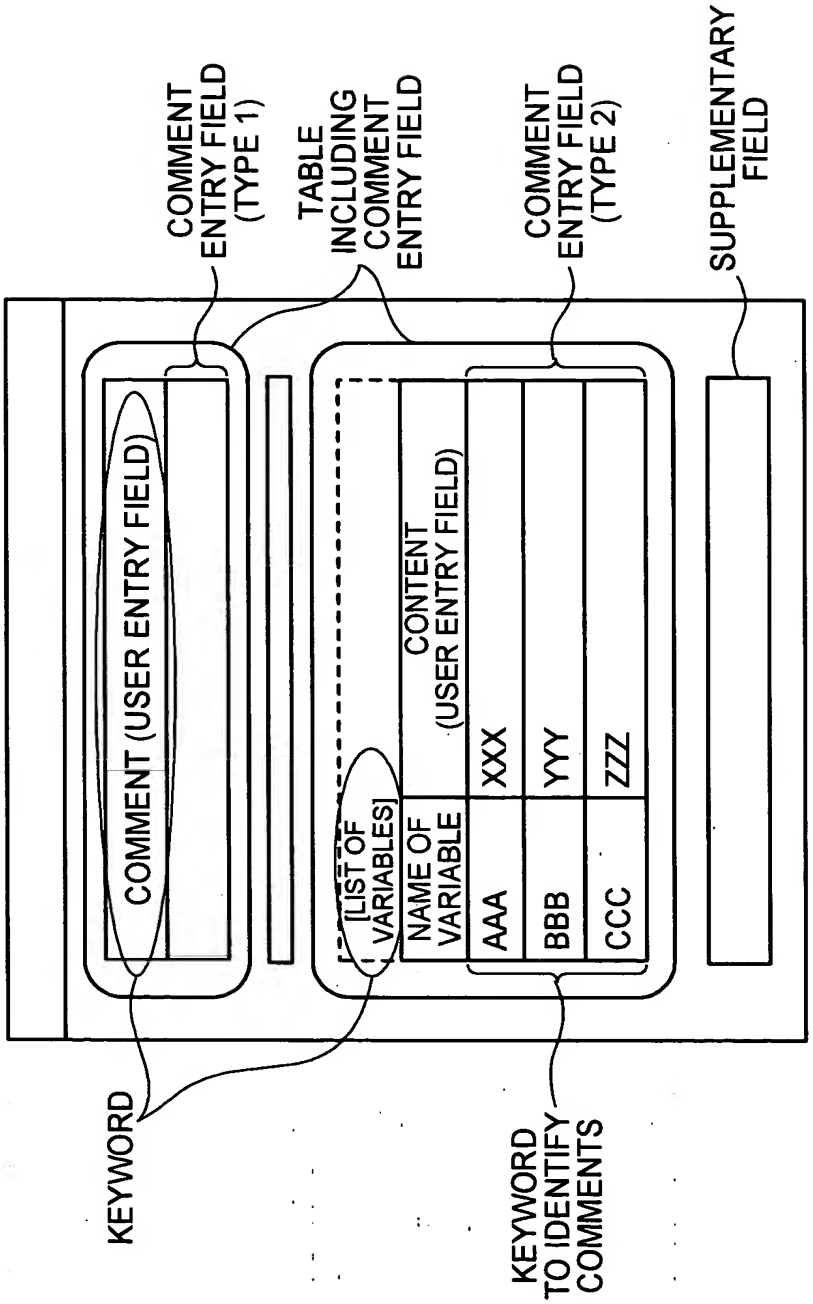

COMMENT ENTRY FIELD (TYPE 1)
TABLE INCLUDING COMMENT ENTRY FIELD
COMMENT ENTRY FIELD (TYPE 2)
SUPPLEMENTARY FIELD
COMMENT (USER ENTRY FIELD)
[LIST OF VARIABLES]
NAME OF VARIABLE
CONTENT (USER ENTRY FIELD)
AAA
BBB
CCC
XXX
YYY
ZZZ
KEYWORD
KEYWORD TO IDENTIFY COMMENTS

FIG.14

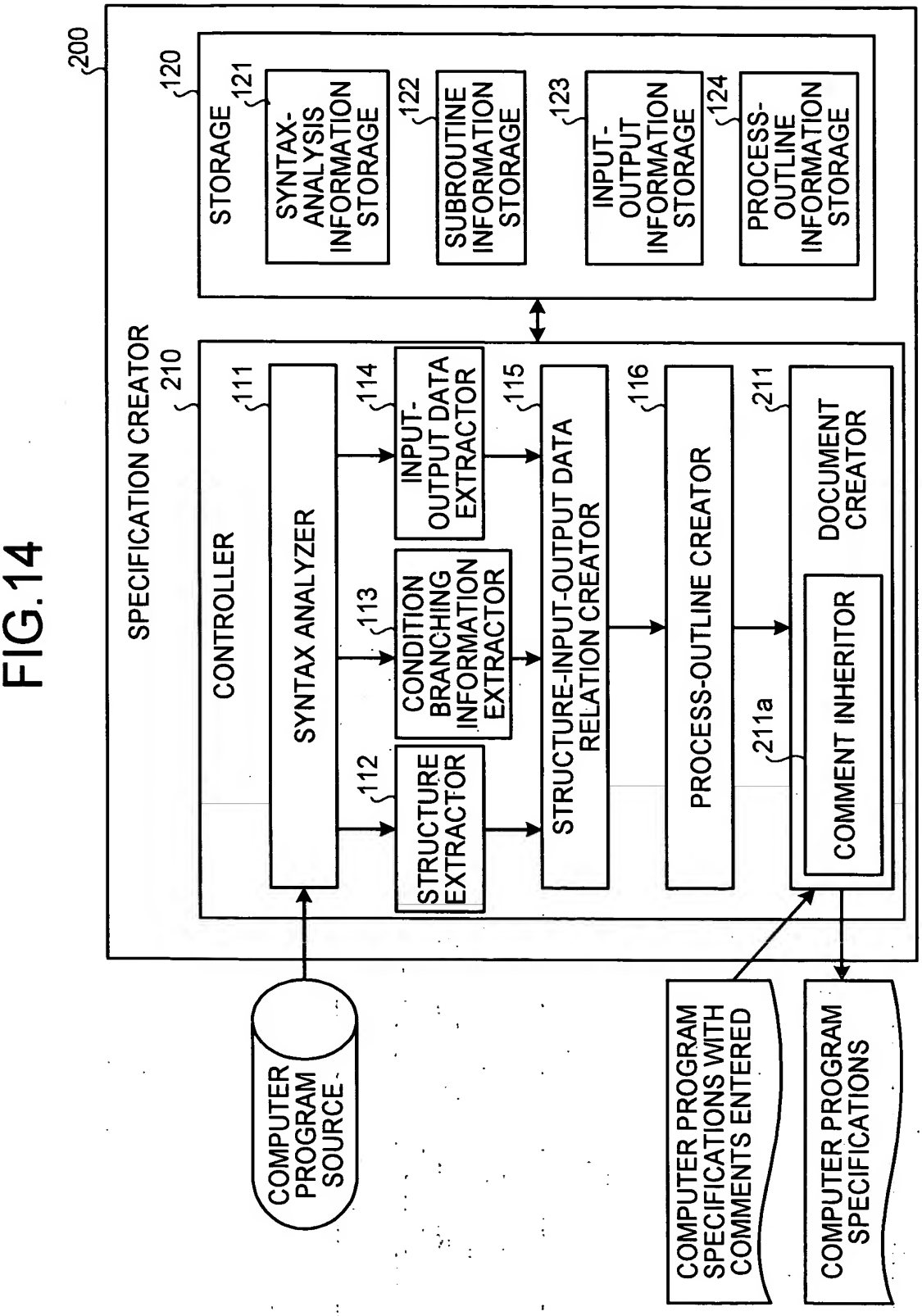

200
SPECIFICATION CREATOR
210
CONTROLLER
111
SYNTAX ANALYZER
112
STRUCTURE EXTRACTOR
113
CONDITION BRANCHING INFORMATION EXTRACTOR
114
INPUT-OUTPUT DATA EXTRACTOR
115
STRUCTURE-INPUT-OUTPUT DATA RELATION CREATOR
116
PROCESS-OUTLINE CREATOR
211
DOCUMENT CREATOR
211a
COMMENT INHERITOR
120
STORAGE
121
SYNTAX-ANALYSIS INFORMATION STORAGE
122
SUBROUTINE INFORMATION STORAGE
123
INPUT-OUTPUT INFORMATION STORAGE
124
PROCESS-OUTLINE INFORMATION STORAGE
COMPUTER PROGRAM SOURCE
COMPUTER PROGRAM SPECIFICATIONS WITH COMMENTS ENTERED
COMPUTER PROGRAM SPECIFICATIONS

FIG.15

Reflection List (FIXED PROPERTY NAME)
Reflection1,Reflection2,...ReflectionN

FIG.16A

Reflection1
KEY WORD

FIG.16B

MultiReflection1
KEYWORD, COMMENTS IDENTIFICATION KEYWORD COLUMN POSITION, COMMENT ENTRY FIELD COLUMN POSITION

FIG.17

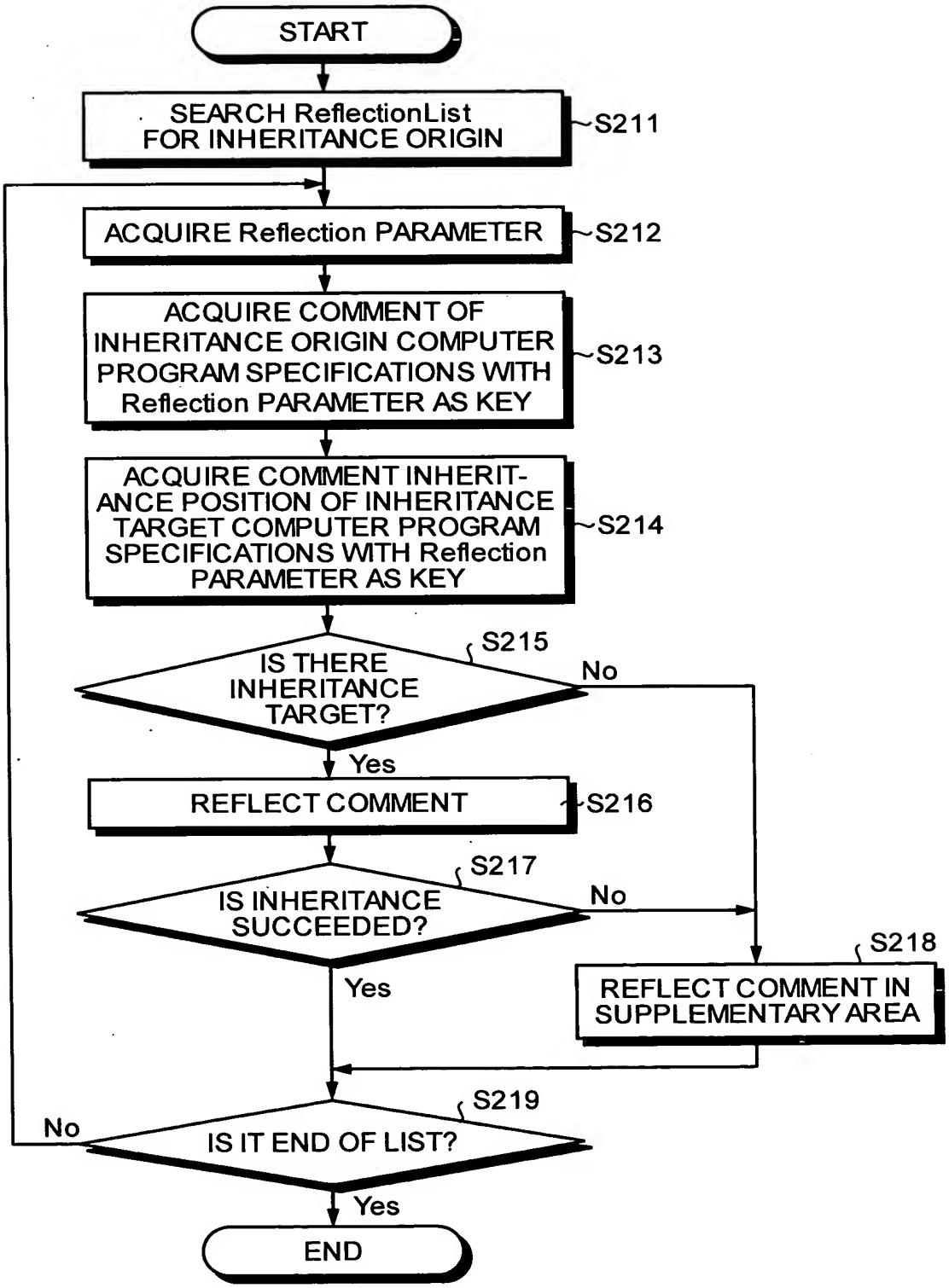

START
SEARCH ReflectionList FOR INHERITANCE ORIGIN
S211
ACQUIRE Reflection PARAMETER
S212
ACQUIRE COMMENT OF INHERITANCE ORIGIN COMPUTER PROGRAM SPECIFICATIONS WITH Reflection PARAMETER AS KEY
S213
ACQUIRE COMMENT INHERIT-ANCE POSITION OF INHERITANCE TARGET COMPUTER PROGRAM SPECIFICATIONS WITH Reflection PARAMETER AS KEY
S214
S215
IS THERE INHERITANCE TARGET?
No
Yes
REFLECT COMMENT
S216
S217
IS INHERITANCE SUCCEEDED?
No
Yes
S218
REFLECT COMMENT IN SUPPLEMENTARY AREA
S219
No
IS IT END OF LIST?
Yes
END

FIG.18

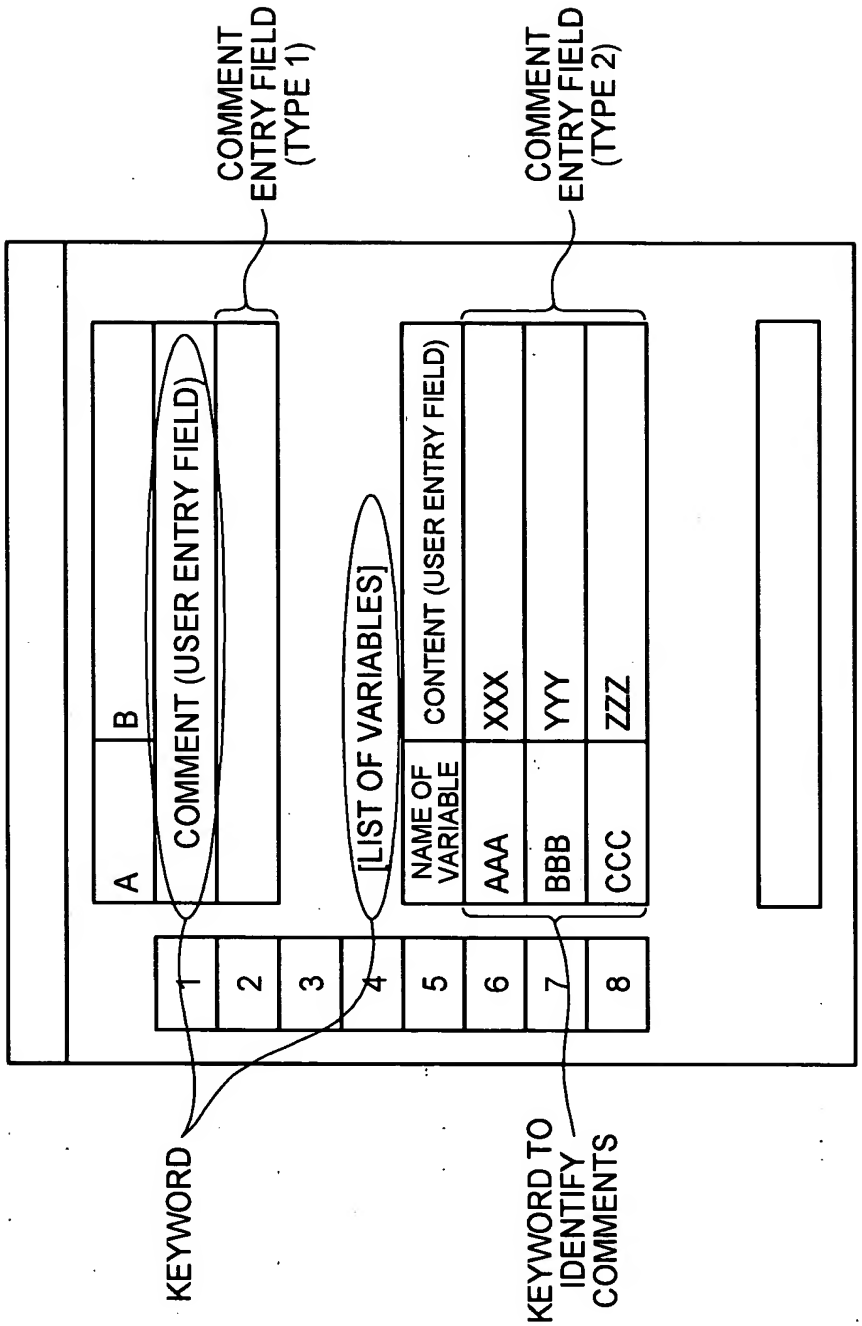

COMMENT ENTRY FIELD (TYPE 1)
COMMENT ENTRY FIELD (TYPE 2)
A
B
COMMENT (USER ENTRY FIELD)
[LIST OF VARIABLES]
NAME OF VARIABLE
CONTENT (USER ENTRY FIELD)
AAA
BBB
CCC
XXX
YYY
ZZZ
1
2
3
4
5
6
7
8
KEYWORD
KEYWORD TO IDENTIFY COMMENTS

FIG.19

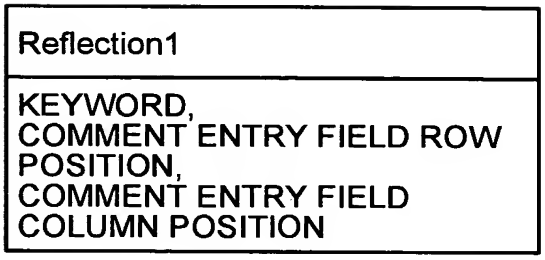

Reflection1
KEYWORD,
COMMENT ENTRY FIELD ROW POSITION,
COMMENT ENTRY FIELD COLUMN POSITION

FIG.20

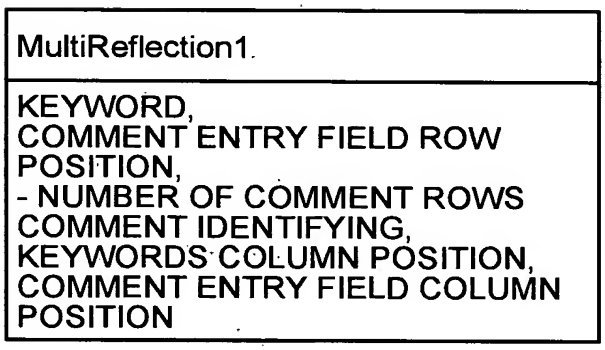

MultiReflection1
KEYWORD,
COMMENT ENTRY FIELD ROW POSITION,
- NUMBER OF COMMENT ROWS
COMMENT IDENTIFYING,
KEYWORDS COLUMN POSITION,
COMMENT ENTRY FIELD COLUMN POSITION

FIG.21

COMPUTER PROGRAM NAME	SECTION (FIRST NESTING LEVEL)	SECTION (SECOND NESTING LEVEL)	READ FILE	WRITE FILE	NOTE
COMPUTER PROGRAM		COMMENT (2)			
	INITIAL PROCESS				
COMMENT (1)					
	EXECUTION CONDITION				
	UNTIL (END FLAG = CONSTANT-END)				
	COMMENT (3)				
	REPEATED PROCESS		in1(rec1)	out1(rec1)	
				out2(rec2)	
				out3(rec3)	
				out4(rec4)	
			COMMENT (5)	out5(rec5)	
		CALL "COM0002"			
		COMMENT (4)			
	REPEATED PROCESS			COMMENT (6)	
		GOTO <loop>			
		<finish>			
		COMMENT (7)			
	END PROCESS				
					COMMENT (8)

FIG.22

INFORMATION NAME→	SECTION STRUCTURE (STRUCTURE CONSIDERIGN NUMBER OF APPEARANCES)	DISPLACEMENT (FROM CORRESPONDING SECTION)	NAME OF ADDED COLUMN (IN A CASE OF COLUMN NOT IN STRUCTURE)
COMMENT (1)	COMPUTER PROGRAM NAME	2 ROWS, 0 COLUMNS	
COMMENT (2)	COMPUTER PROGRAM NAME	0 ROWS, 2 COLUMNS	
COMMENT (3)	COMPUTER PROGRAM NAME └INITIAL PROCESS [1]	4 ROWS, 0 COLUMNS	
COMMENT (4)	COMPUTER PROGRAM NAME └REPEATED PROCESS [1] └CALL "COM0002" [1]	1 ROWS, 0 COLUMNS	
COMMENT (5)	COMPUTER PROGRAM NAME └REPEATED PROCESS [1]		FILE TO BE READ
COMMENT (6)	COMPUTER PROGRAM NAME └REPEATED PROCESS [2]	2 ROWS, 0 COLUMNS	FILE TO BE WRITTEN
COMMENT (7)	COMPUTER PROGRAM NAME └REPEATED PROCESS [2] └<finish> [1]		
COMMENT (8)	COMPUTER PROGRAM NAME └END PROCESS [1]		NOTE

FIG.23

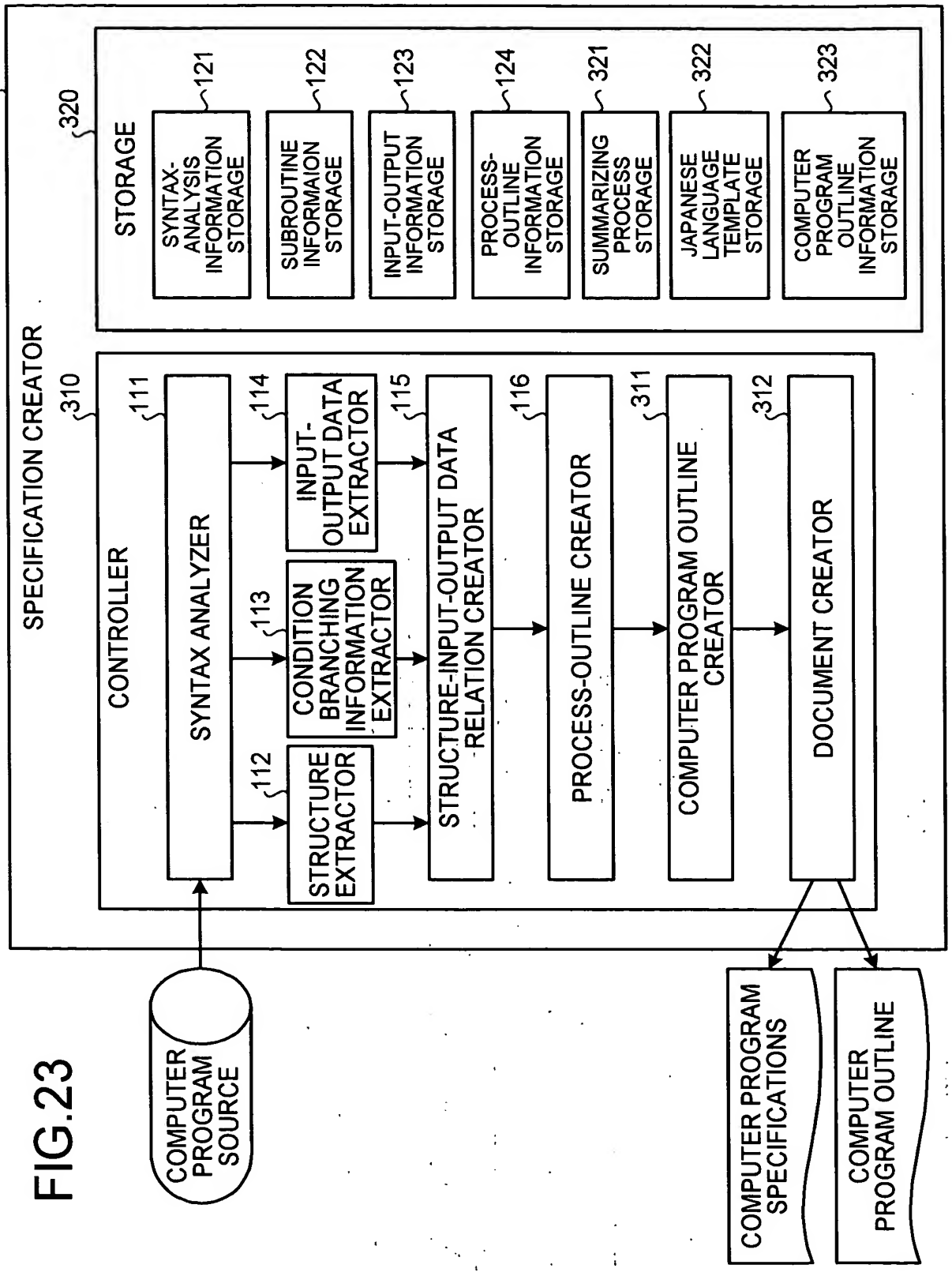

300
SPECIFICATION CREATOR
320
STORAGE
121
SYNTAX-ANALYSIS INFORMATION STORAGE
122
SUBROUTINE INFORMAION STORAGE
123
INPUT-OUTPUT INFORMATION STORAGE
124
PROCESS-OUTLINE INFORMATION STORAGE
321
SUMMARIZING PROCESS STORAGE
322
JAPANESE LANGUAGE TEMPLATE STORAGE
323
COMPUTER PROGRAM OUTLINE INFORMATION STORAGE
310
CONTROLLER
111
SYNTAX ANALYZER
112
STRUCTURE EXTRACTOR
113
CONDITION BRANCHING INFORMATION EXTRACTOR
114
INPUT-OUTPUT DATA EXTRACTOR
115
STRUCTURE-INPUT-OUTPUT DATA RELATION CREATOR
116
PROCESS-OUTLINE CREATOR
311
COMPUTER PROGRAM OUTLINE CREATOR
312
DOCUMENT CREATOR
COMPUTER PROGRAM SOURCE
COMPUTER PROGRAM SPECIFICATIONS
COMPUTER PROGRAM OUTLINE

FIG.24

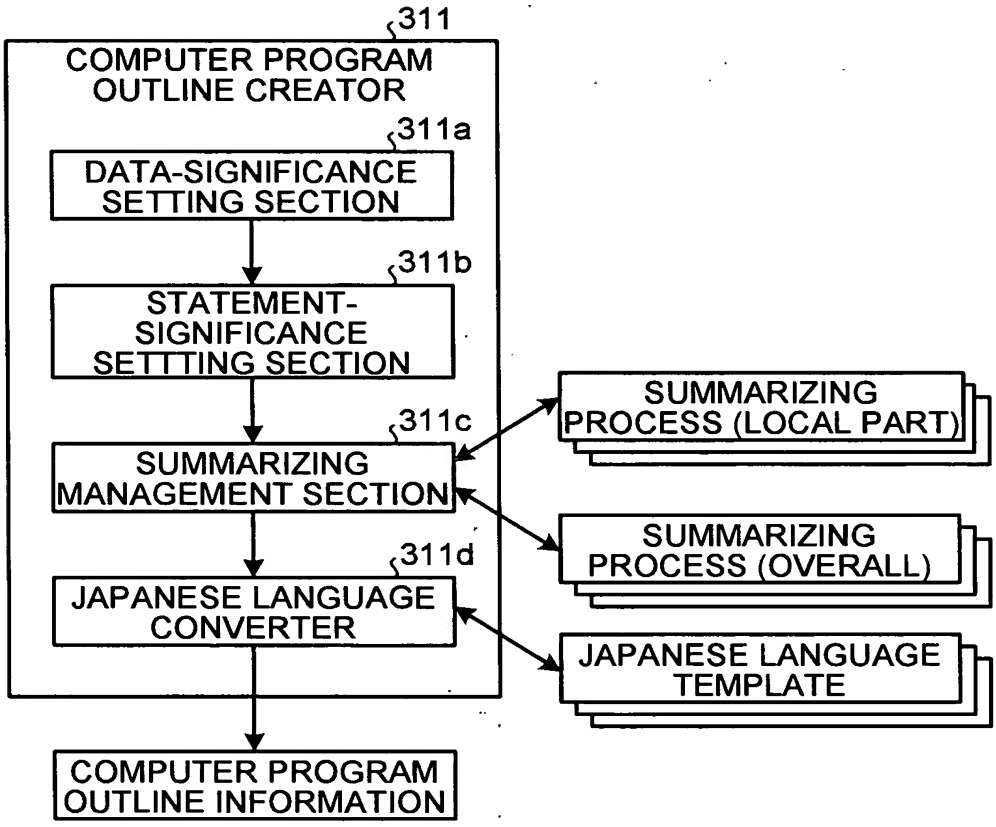

311
COMPUTER PROGRAM OUTLINE CREATOR
311a
DATA-SIGNIFICANCE SETTING SECTION
311b
STATEMENT-SIGNIFICANCE SETTTING SECTION
311c
SUMMARIZING MANAGEMENT SECTION
311d
JAPANESE LANGUAGE CONVERTER
COMPUTER PROGRAM OUTLINE INFORMATION
SUMMARIZING PROCESS (LOCAL PART)
SUMMARIZING PROCESS (OVERALL)
JAPANESE LANGUAGE TEMPLATE

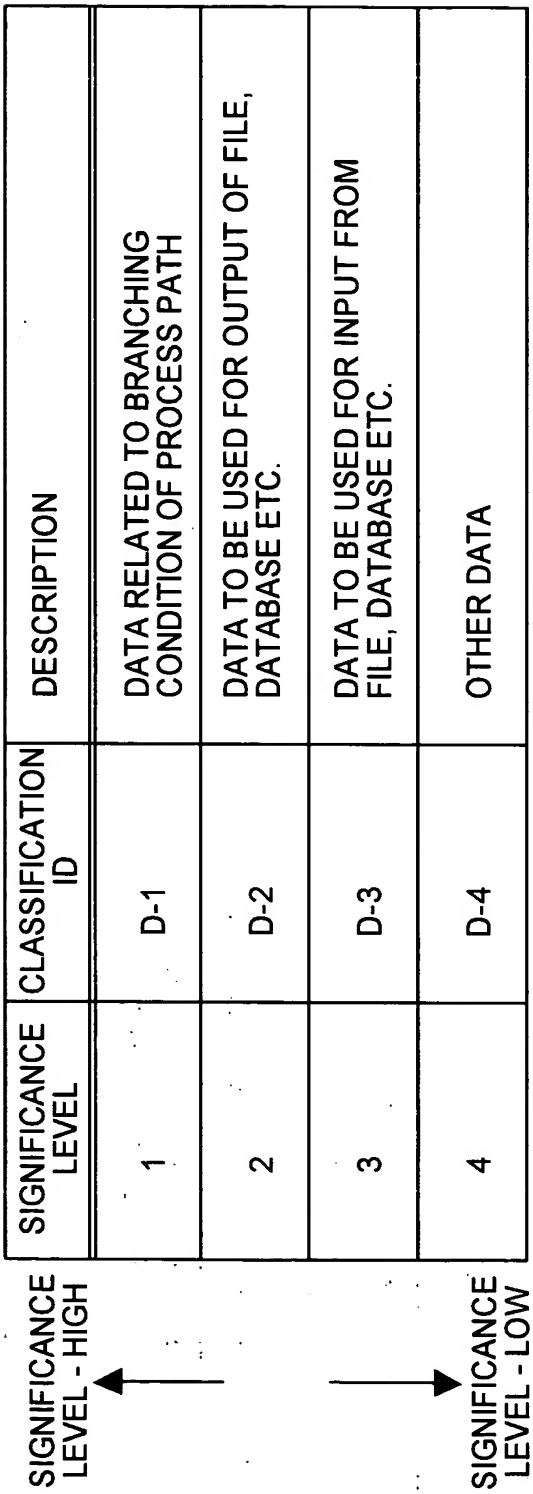
FIG.25A

```
MAIN                    SECTION.
MAIN-START.

     MOVE 0 TO W-ERR-FLAG.
     IF  IN-CODE  NOT  =  103
        GO TO   NEXT-DATA-READ
     END-IF.

     PERFORM   MASTER-MODIFY-SECT.
     IF  W-ERR-FLAG  NOT  =  0
         GO TO   MAIN-END
     END-IF.

NEXT-DATA-READ.
     PERFORM   FILE-READ-SECT.

MAIN-END.
     EXIT.
```

FIG.25B

```
< section name="MAIN">
   <paragraph name="MAIN-START">
      <sequences num="4">
        <move>
              <ref><constant value="0" type="int"/></ref>
          <def><var name="W-ERR-FLAG"/></def>
        </move>
        <if>
            <condition><expression>
              <var name="IN-CODE"/>
                <comparison_operator name="NOT ="/>
                  <constant value="103" type="int"/>
              </expression></condition>
            <then>
              <sequences num="1">
                <goto>
                    <target type="paragraph"
                          name="NEXT-DATA-READ"
                            section_name="MAIN"/>
                </goto>
              </sequences>
            </then>
        </if>
         <perform_external>
            <target type="section" name="MASTER-MODIFY-SECT"/>
          </perform_external>
        <if>
            <condition><expression>
              <var name="W-ERR-FLAG"/>
                <comparison_operator name="NOT ="/>
                  <figurative_constant name="ZERO"/>
              </expression></condition>
            <then>
              <sequences num="1">
                <goto>
                    <target type="paragraph"
                          name="MAIN-END"
                            section_name="MAIN"/>
                </goto>
              </sequences>
            </then>
        </if>
      </sequences>
   </paragraph>
   <paragraph name="NEXT-DATA-READ">
      <sequences num="1">
         <perform_external>
              <target type="section" name="FILE-READ-SECT"/>
          </perform_external>
      </sequences>
   </paragraph>
   <paragraph name="MAIN-END">
      <sequences num="1">
         <exit_sentence/>
      </sequences>
   </paragraph>
 </section>
```

FIG.26



SIGNIFICANCE LEVEL	CLASSIFICATION ID	DESCRIPTION
1	D-1	DATA RELATED TO BRANCHING CONDITION OF PROCESS PATH
2	D-2	DATA TO BE USED FOR OUTPUT OF FILE, DATABASE ETC.
3	D-3	DATA TO BE USED FOR INPUT FROM FILE, DATABASE ETC.
4	D-4	OTHER DATA

FIG.27

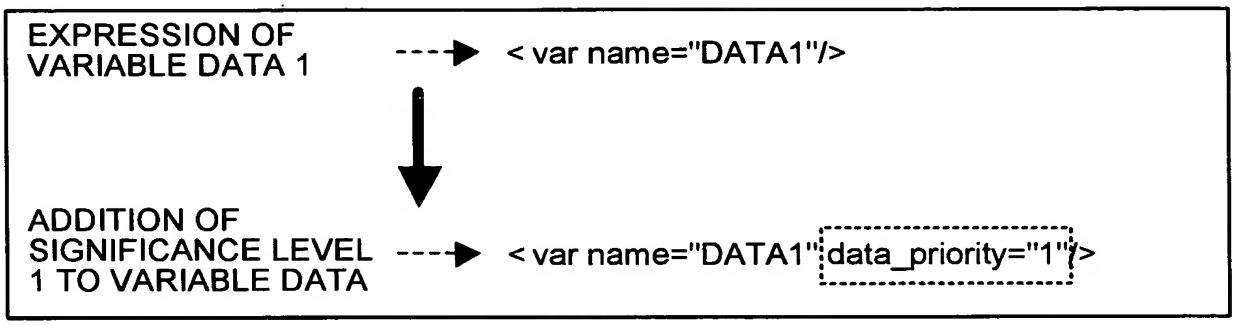

EXPRESSION OF VARIABLE DATA 1
< var name="DATA1"/>
ADDITION OF SIGNIFICANCE LEVEL 1 TO VARIABLE DATA
< var name="DATA1" data_priority="1"/>

FIG.28

SIGNIFICANCE LEVEL	CLASSIFICATION ID	DESCRIPTION
1	S-1	FOLLOWING STATEMENTS THAT PERFORM FOLLOWING OPERATIONS FOR DATA THAT CORRESPONDS TO SIGNIFICANCE LEVEL D-1 AND D-2. STATEMENT THAT REWRITES SUBSTITUTION ETC., STATEMENT THAT CALLS SUBROUTINE, AND STATEMENT THAT INPUTS AND OUTPUTS TO FILE, DATABASE
2	S-2	CONDITION-JUDGMENT STATEMENT THAT INCLUDES STATEMENT CLASSIFIED AS S-1 AS STATEMENT TO BE EXECUTED IMMEDIATELY AFTER JUDGMENT.
3	S-3	OTHER STATEMENTS

FIG.29

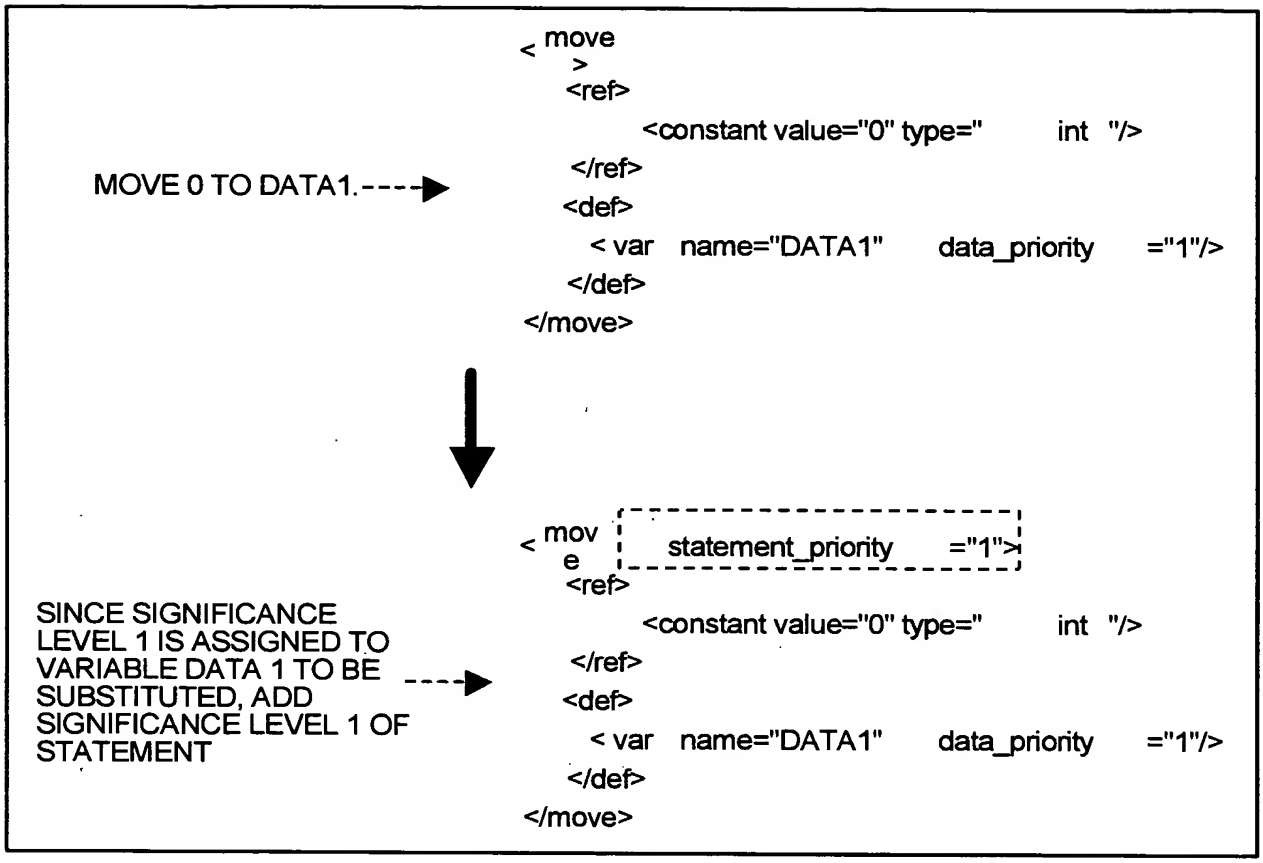

MOVE 0 TO DATA1.
< move
>
<ref>
<constant value="0" type=" int "/>
</ref>
<def>
< var name="DATA1" data_priority ="1"/>
</def>
</move>
SINCE SIGNIFICANCE LEVEL 1 IS ASSIGNED TO VARIABLE DATA 1 TO BE SUBSTITUTED, ADD SIGNIFICANCE LEVEL 1 OF STATEMENT
< mov
e
statement_priority ="1">
<ref>
<constant value="0" type=" int "/>
</ref>
<def>
< var name="DATA1" data_priority ="1"/>
</def>
</move>

FIG.30

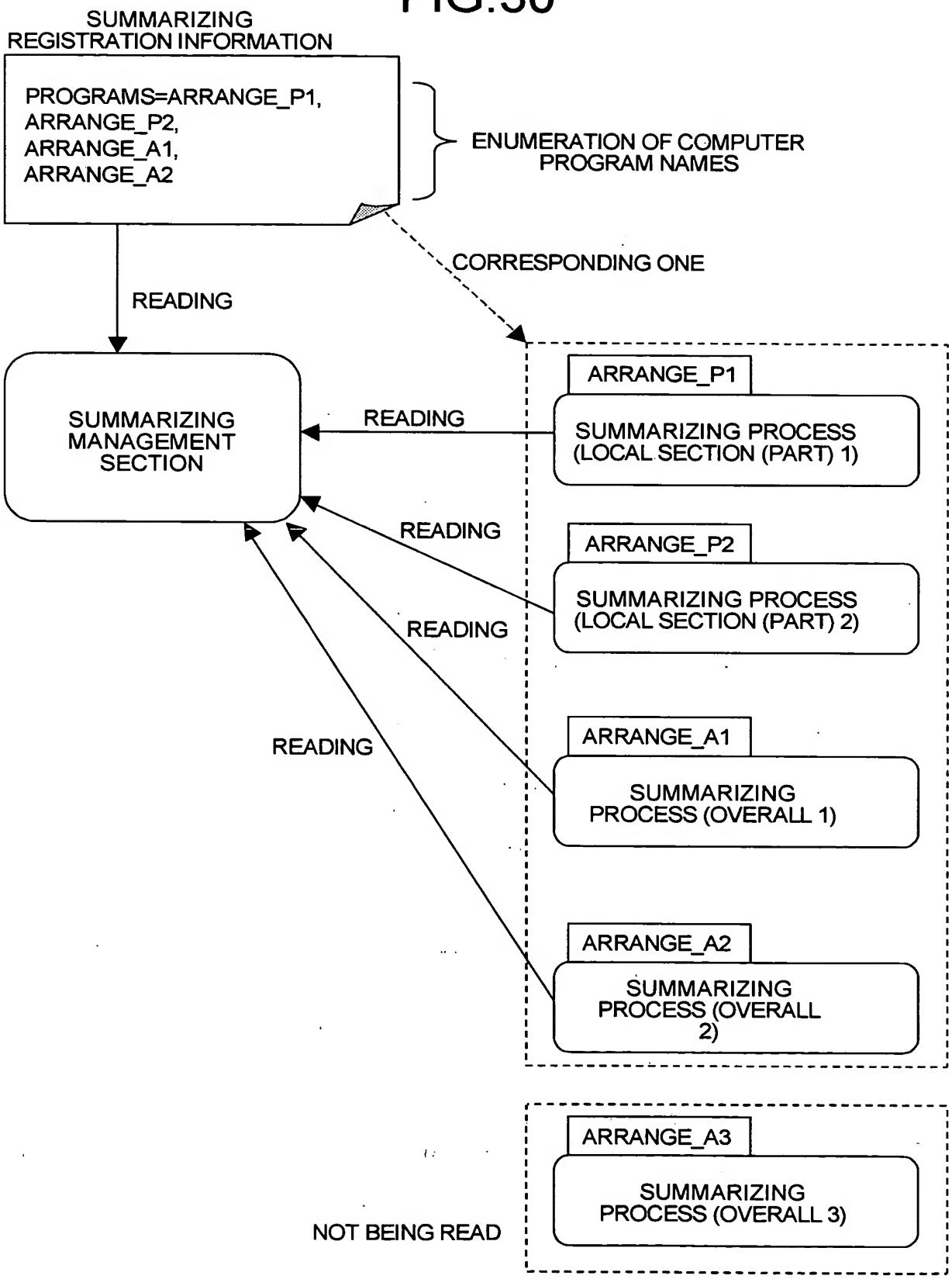

SUMMARIZING REGISTRATION INFORMATION
PROGRAMS=ARRANGE_P1,
ARRANGE_P2,
ARRANGE_A1,
ARRANGE_A2
ENUMERATION OF COMPUTER PROGRAM NAMES
CORRESPONDING ONE
READING
SUMMARIZING MANAGEMENT SECTION
READING
READING
READING
READING
ARRANGE_P1
SUMMARIZING PROCESS (LOCAL SECTION (PART) 1)
ARRANGE_P2
SUMMARIZING PROCESS (LOCAL SECTION (PART) 2)
ARRANGE_A1
SUMMARIZING PROCESS (OVERALL 1)
ARRANGE_A2
SUMMARIZING PROCESS (OVERALL 2)
ARRANGE_A3
SUMMARIZING PROCESS (OVERALL 3)
NOT BEING READ

FIG.31

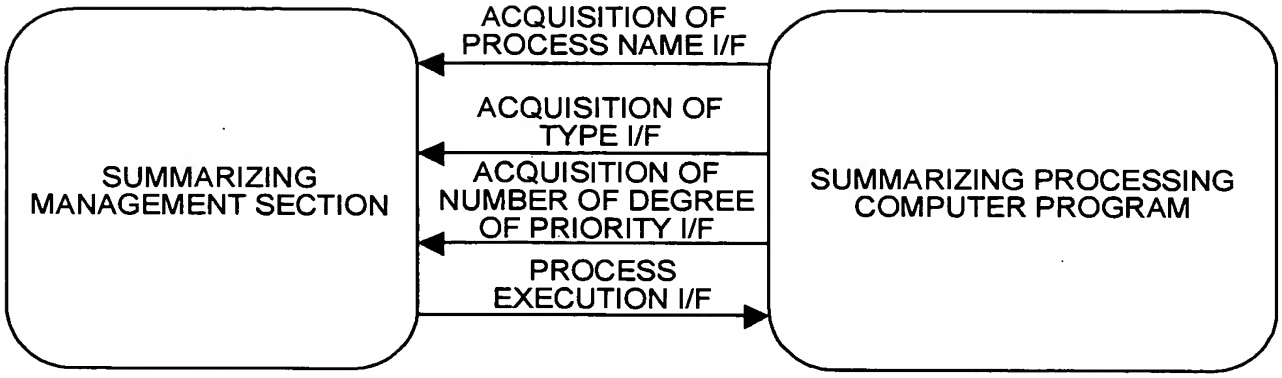

SUMMARIZING MANAGEMENT SECTION
ACQUISITION OF PROCESS NAME I/F
ACQUISITION OF TYPE I/F
ACQUISITION OF NUMBER OF DEGREE OF PRIORITY I/F
PROCESS EXECUTION I/F
SUMMARIZING PROCESSING COMPUTER PROGRAM

FIG.32

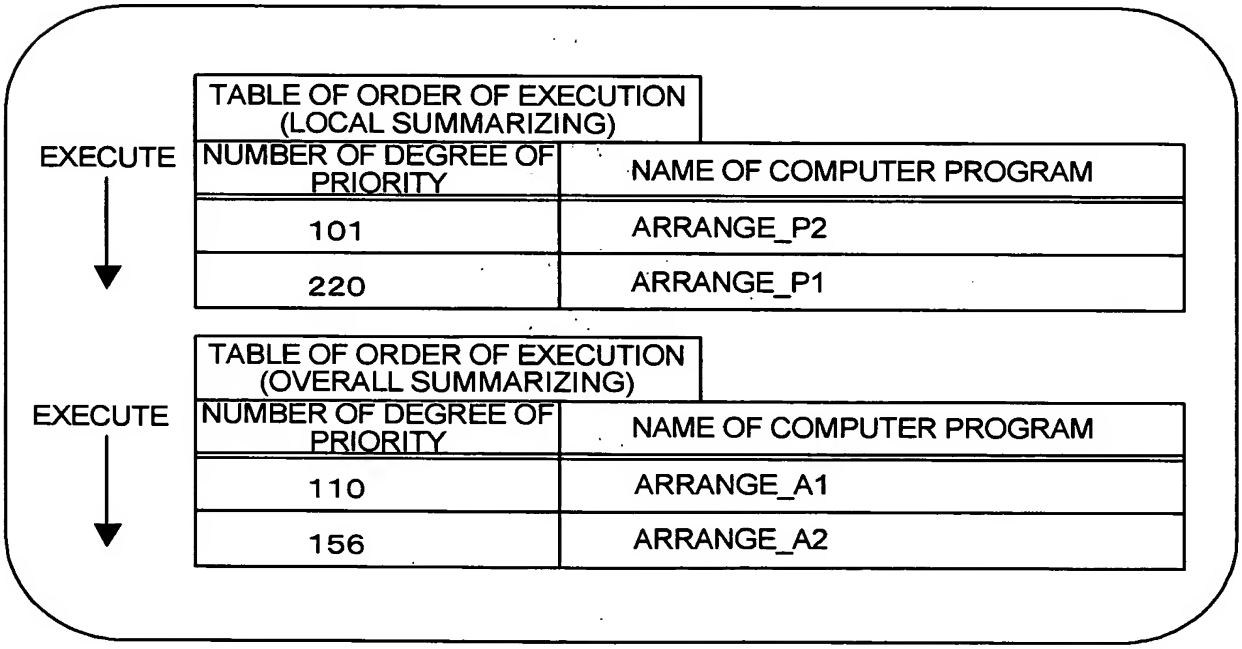

EXECUTE
TABLE OF ORDER OF EXECUTION (LOCAL SUMMARIZING)
NUMBER OF DEGREE OF PRIORITY
NAME OF COMPUTER PROGRAM
101
ARRANGE_P2
220
ARRANGE_P1
EXECUTE
TABLE OF ORDER OF EXECUTION (OVERALL SUMMARIZING)
NUMBER OF DEGREE OF PRIORITY
NAME OF COMPUTER PROGRAM
110
ARRANGE_A1
156
ARRANGE_A2

FIG.33A

NAME OF COMPUTER PROGRAM:AR_P1	PROCESS NAME: SUMMARIZING OF READ-UNTIL
TYPE: LOCAL	NUMBER OF DEGREE OF PRIORITY: 100

DESCRIPTION:

IF A PROCESS AT THE END OF FILE OF "READ" STATEMENT MATCHES WITH SATISFYING OF "UNTIL" CONDITION, REPLACE CONDITION OF "UNTIL" BY 'TILL THE END OF FILE'. DELETE PROCESS UP TO FILE END OF "READ" STATEMENT.

EXAMPLE:

```
READ INF AT END MOVE "END" TO FLAG .

PERFORM ~ UNTIL FLAG = "END".
```

```
READ INF.

PERFORM ~ UNTIL AT_FILE_END INF.
```

TAG <at_file_end> IS INSERTED IN PROCESS.

FIG.33B

NAME OF COMPUTER PROGRAM:AR_P2	PROCESS NAME: DEVELOPMENT OF SECTION
TYPE: LOCAL	NUMBER OF DEGREE OF PRIORITY: 200

DESCRIPTION:

IF CONTENT OF SECTION IS SHORT, UP TO 3 ROWS AND IF IT IS CALLED WITHOUT ANY CONDITION IN ORIGIN OF CALLING (HIGHER RANK SECTION), DEVELOP CONTENT OF SECTION SUCH THAT IT IS INCLUDED IN HIGHER RANK SECTION.

EXAMPLE:

```
~
   PERFORM  READ   - SUB.
~
READ -  SUB          SECTION.
  READ INF .
  EXIT.
```

```
~
  READ INF.
~
```

FIG.33C

NAME OF COMPUTER PROGRAM:AR_P3	PROCESS NAME: GROUP ITEMIZATION OF SUBSTITUTION (SUBSTITUTE)
TYPE: LOCAL	NUMBER OF DEGREE OF PRIORITY: 300

DESCRIPTION:

IF MOVE STATEMENT CONTINUES, ALL ITEMS OF ORIGIN OF SUBSTITUTION BELONG TO SAME GROUP ITEM, AND ALL ITEMS OF TARGET OF SUBSTITUTION BELONG TO SAME GROUP ITEM, REPLACE THEM TO SUBSTITUTE STATEMENT BY NAMES OF GROUP ITEMS. IF NOT ALL ITEMS IN GROUP ITEMS ARE ENUMERATED, ADD ATTRIBUTE THAT INDICATES BEING A PART OF THAT GROUP ITEM.

EXAMPLE:

```
MOVE IN-DATA1 TO OUT-DATA1.
MOVE IN-DATA2 TO OUT-DATA1.
(IN-DATA1, IN-DATA2 BELONG TO INR,
OUT0DATA1, OUT-DAT2 BELONG TO OUTR)
```

```
MOVE INR TO OUTR.
```

FIG.33D

NAME OF COMPUTER PROGRAM:AR_P4		PROCESS NAME: DELETION OF STATEMENTS WITH LOW SIGNIFICANCE LEVEL
TYPE: LOCAL	NUMBER OF DEGREE OF PRIORITY: 400	

DESCRIPTION:

(1) DELETE STATEMENT FOR WHICH statement_priority IS 3. WHILE DELETING, REDUCE VALUE OF ATTRIBUTE qt OF PARENT TAG <sequence> OF TAG OF STATEMENT THAT IS TO BE DELETED BY NUMBER OF STATEMENTS THAT ARE DELETED. (qt INDICATES NUMBER OF STATEMENTS INSIDE).

(2) WHILE DELETING, IF VALUE OF ATTRIBUTE qt OF <sequence> IS 0, DELETE SECTION, PARAGRAPH INCLUDING THE VALUE OR CONDITION STATEMENT (CONDITION AFTER IF, EVALUATE, READ).

PERFORM (1) AND (2) REPEATEDLY TILL NUMBER OF STATEMENTS IN COMPUTER PROGRAM STOPS CHANGING.

FIG.33E

NAME OF COMPUTER PROGRAM:AR_A1	PROCESS NAME: DETECTION OF LOOP OF READ
TYPE: OVERALL	NUMBER OF DEGREE OF PRIORITY: 100

DESCRIPTION:

IF THERE IS A PROCESS OF FILE READING, A LOOP AFTER THE PROCESS OF FILE READING, AND IF A FILE IS READ AT THE END OF A PROCESS INSIDE THE LOOP, REPLACE IT TO 'READ FILE REPEATEDLY TILL CONDITION OF LOOP IS SATISFIED'. IF LOOP IS A SECTION, DEVELOP IT TO A PART THAT IS CALLING THE SECTION.

EXAMPLE:

```
PROCESS 1
READ INF.
PERFORM SUB1 UNTIL CONDITION A.
PROCESS 2

SUB1   SECTION.
   PROCESS 3
   READ INF.
   EXIT.
```

```
PROCESS 1
LOOP-READ INF UNTIL
CONDITION A
PROCESS 3
END-LOOP-READ
PROCESS 2
```

SET A NEW STATEMENT. IN PROCESS, PREPARE <loop_read_file> TAG NEWLY.

FIG.33F

NAME OF COMPUTER PROGRAM:AR_A2		PROCESS NAME: ENTRY OF OVERALL NAME OF VARIABLE
TYPE: OVERALL	NUMBER OF DEGREE OF PRIORITY: 200	

DESCRIPTION:

EXPRESS ITEM NAME OF A VARIABLE WITH GROUP ITEM NAME ADDED TO IT. LINK BY USING "." BETWEEN THE NAMES. EXPRESS FROM THE HIGHEST GROUP ITEM NAME.

FORMAT OF IN-RECORD.IN-DATA1 ETC.

FIG.34

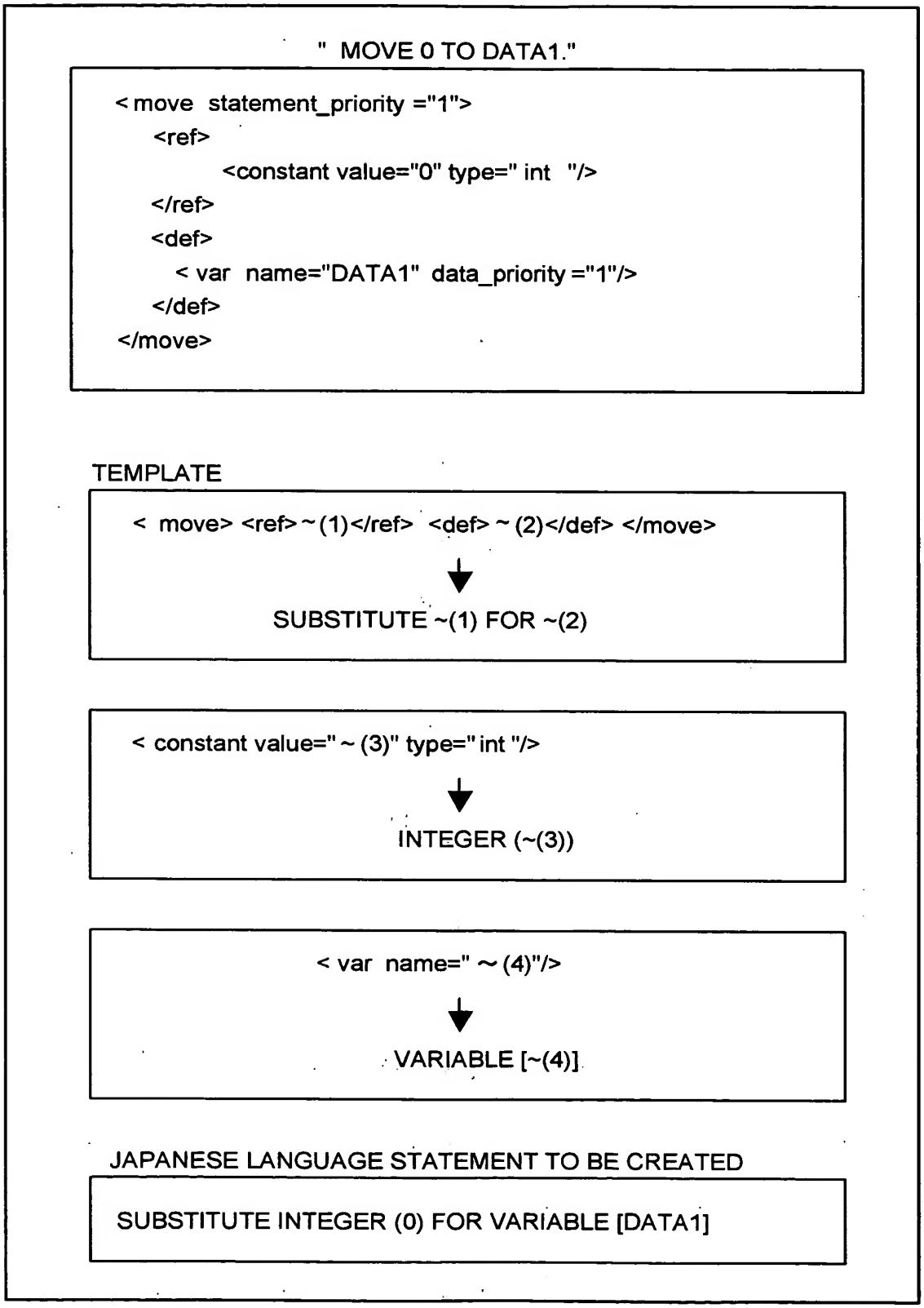

" MOVE 0 TO DATA1."
< move statement_priority ="1">
<ref>
<constant value="0" type=" int "/>
</ref>
<def>
< var name="DATA1" data_priority ="1"/>
</def>
</move>
TEMPLATE
< move> <ref> ~ (1)</ref> <def> ~ (2)</def> </move>
SUBSTITUTE ~(1) FOR ~(2)
< constant value=" ~ (3)" type=" int "/>
INTEGER (~(3))
< var name=" ~ (4)"/>
VARIABLE [~(4)]
JAPANESE LANGUAGE STATEMENT TO BE CREATED
SUBSTITUTE INTEGER (0) FOR VARIABLE [DATA1]

FIG.35A

```
< loop_read_file >
    <file name=" ~ (1)">
      <record> ~ (2)</record>
    </file>
    <until> ~ (3)</until>
  <sequences> ~ (4)</sequences>
</ loop_read_file >
```

↓

READ RECORD FROM FILE ~(1) TO ~(2) AND PERFORM UP TO ~(3) REPEATEDLY.
PERFORM FOLLOWING WHEN READ EVERY TIME
~(4)

```
< if>
    <condition> ~ (1)</condition>
  <then><sequence>~ (2)</sequence></then>
  <else><sequence>~ (3)</sequence></else>
</if>
```

↓

IF CONDITION ~(1) IS SATISFIED, PERFORM FOLLOWING ~(2)

IF CONDITION ~(1) IS NOT SATISFIED, PERFORM FOLLOWING ~(3)

```
< write>
      <file name=" ~ (1)">
        <record>~ (2)</record>
      </file>
</write>
```

↓

WRITE RECORD ~(2) IN FILE ~(1).

FIG.35B

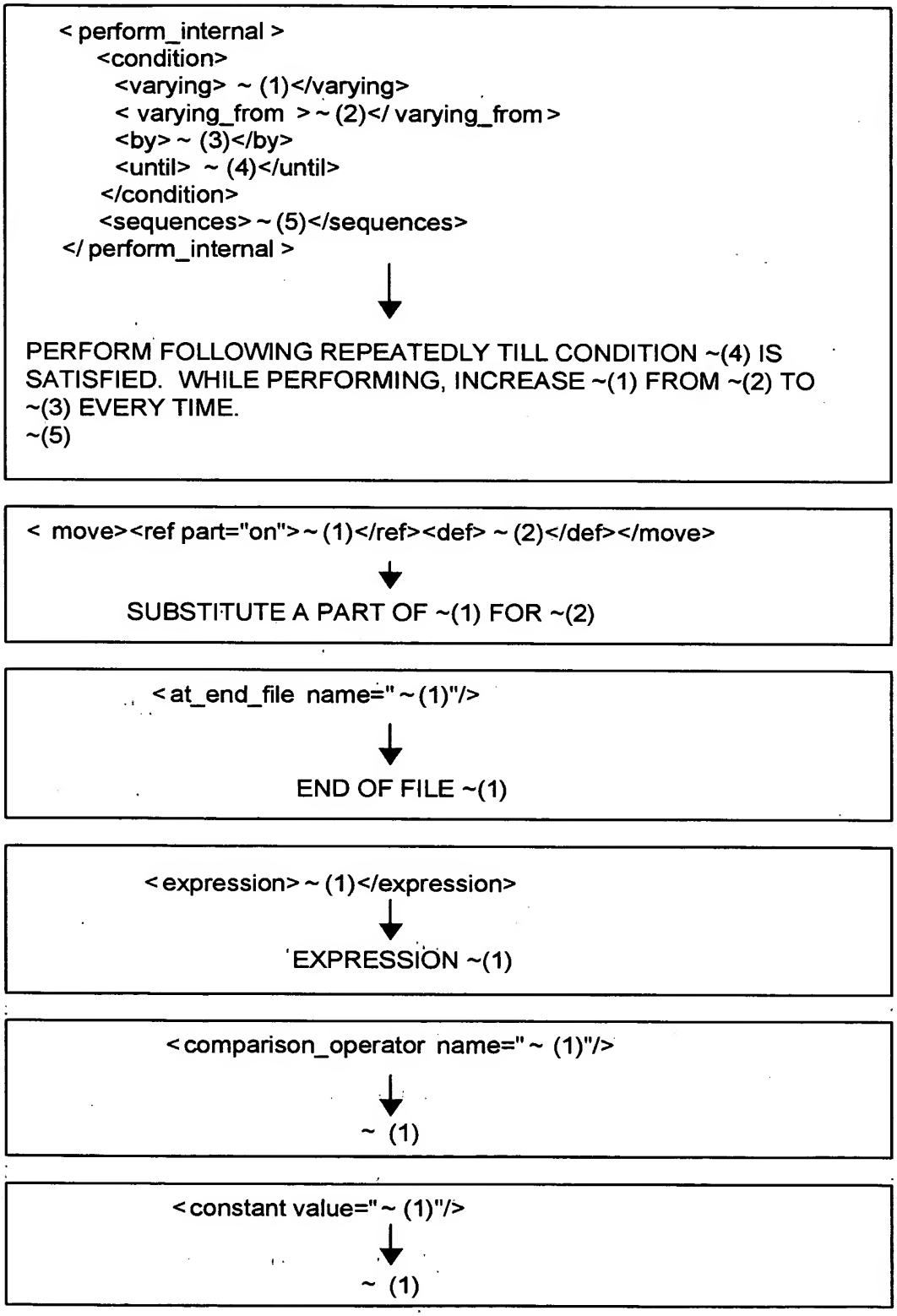

<perform_internal>
<condition>
<varying> ~ (1)</varying>
< varying_from > ~ (2)</ varying_from >
<by> ~ (3)</by>
<until> ~ (4)</until>
</condition>
<sequences> ~ (5)</sequences>
</ perform_internal >
PERFORM FOLLOWING REPEATEDLY TILL CONDITION ~(4) IS SATISFIED. WHILE PERFORMING, INCREASE ~(1) FROM ~(2) TO ~(3) EVERY TIME.
~(5)
< move><ref part="on">~ (1)</ref><def> ~ (2)</def></move>
SUBSTITUTE A PART OF ~(1) FOR ~(2)
<at_end_file name="~ (1)"/>
END OF FILE ~(1)
<expression>~ (1)</expression>
EXPRESSION ~(1)
<comparison_operator name="~ (1)"/>
~ (1)
<constant value="~ (1)"/>
~ (1)

FIG.36

```
IDENTIFICATION              DIVISION.
PROGRAM-ID.                 TESTSAMPLE.
AUTHOR.                     TARO.YAMADA.
ENVIRONMENT                 DIVISION.
CONFIGURATION               SECTION.
SOURCE-COMPUTER.            VIRTUAL/HOST1.
OBJECT-COMPUTER.            VIRTUAL/HOST1.

   ********** IN OUT **********
INPUT-OUTPUT          SECTION.
FILE-CONTROL.
    SELECT INFILE     ASSIGN TO INFILE.
    SELECT OUTFILE    ASSIGN TO OUTFILE.

  ********** DATA DIVISION **********
DATA DIVISION.
FILE SECTION.

  ********** INPUT FILE **********
  FD  INFILE   BLOCK 0 RECORDS.
01  IN-RECORD.
    03  MEMBERCODE         PIC 9(5).
    03  JOB-CODE           PIC 9(3).
    03  NAME.
        05  NAMEFAMLY      PIC N(16).
        05  NAMEFIRST      PIC X(16).
    03  SEIBETSU           PIC 9.
    03  YUUBINBANGO.
        05  YUBIGCODE      PIC 9(3).
        05  YUSMALLCODE.   PIC 9(4).
    03  TEL                PIC X(14).
    03  BIRTHDAY.
        05  BD-YEAR        PIC 9(4).
        05  BD-MONTH       PIC 9(2).
        05  BD-DAY         PIC 9(2).
    03  FAMILYDATA.
        05  F-MEMBER       OCCURS 20.
          07  F-TYPE       PIC 9(2).
          07  F-MEMBER-F   PIC 9.
          07  F-CODE       PIC 9(5).
    03  FILLER             PIC X(25).

  ********** OUTPUT FILE **********
  FD  OUTFILE    BLOCK 0 RECORDS.
01  OUT-RECORD.
    03  OUT-MEMBERCODE     PIC 9(5).
    03  OUT-F-TYPE         PIC 9(2).
    03  OUT-MEM-FLAG       PIC 9.
    03  OUT-F-CODE         PIC 9(5).
  ********** WORKING STORAGE **********
WORKING-STORAGE SECTION.
01  COUNTER-TABLE.
    03  IN-COUNT           PIC S9(9)  VALUE ZERO.
    03  SKIP-COUNT         PIC S9(9)  VALUE ZERO.
    03  PROC-COUNT         PIC S9(9)  VALUE ZERO.
    03  OUT-COUNT          PIC S9(9)  VALUE ZERO.
  01  I                    PIC S9(4).
01  END-FLAG               PIC X(03)  VALUE SPACE.
```

```
************ PROCEDURE DIVISION ****************************
PROCEDURE DIVISION.

    OPEN   INPUT  INFILE    OUTPUT  OUTFILE.
 DISPLAY "## COMPUTER PROGRAM START UPON CONSOLE.

 PERFORM  READ-SECT.
 IF END-FLAG  = "END"
      DISPLAY "## NOT EVEN ONE CAN READ RECORD."
      UPON CONSOLE
      DISPLAY "## ERROR ENDED."  UPON CONSOLE
      STOP RUN
 END-IF.                                              [1]

 PERFORM  MAIN-SECT  UNTIL  END-FLAG = "END"

   CLOSE  INFILE    OUTFILE.
  DISPLAY "NUMBER OF OUTPUTS =["  OUT-COUNT "]"
  UPON CONSOLE.
  DISPLAY "ENDED NORMALLY."  UPON CONSOLE.

  STOP RUN.

   ********** READ ROUTINE **************************************
READ-SECT   SECTION.
   READ   INFILE
       AT END      MOVE "END" TO  END-FLAG
       NOT AT END  ADD  1   TO  IN-COUNT
   END-READ.                                          [2]
READ-SECT-END
   EXIT.

   ********** MAIN ROUTINE **************************************
MAIN-SECT   SECTION.
      IF FAMILYDATA = ZERO
          ADD   1     TO    SKIP-COUNT
      ELSE
          ADD   1     TO    PROCCOUNT
          PERFORM VARYING I FROM 1 BY 1
                                UNTIL I > 20
            MOVE  MEMBERCODE  TO OUT-MEMBERCODE
            MOVE  F-TYPE(I)         TO OUT-F-TYPE
 [3]        MOVE  F-MEMBER-F(I) TO OUT-F-MEM-FLAG
            MOVE  F-CODE(I)         TO OUT-F-CODE
            WRITE  OUT-RECORD
             ADD   1          TO OUT-COUNT
          END-PERFORM
      END-IF.

      PERFORM  READ-SECT.

MAIN-SECT-END.
      EXIT.
```

FIG.37

NAME OF COMPUTER PROGRAM: AR_P1	PROCESS NAME: SUMMARIZING OF READ-UNTIL	TYPE: LOCAL	NUMBER OF DEGREE OF PRIORITY: 100
NAME OF COMPUTER PROGRAM: AR_P2	PROCESS NAME: DEVELOPMENT OF SECTION	TYPE: LOCAL	NUMBER OF DEGREE OF PRIORITY: 200
NAME OF COMPUTER PROGRAM: AR_P3	PROCESS NAME: GROUP ITEMIZATION OF SUBSTITUTION	TYPE: LOCAL	NUMBER OF DEGREE OF PRIORITY: 300
NAME OF COMPUTER PROGRAM: AR_P4	PROCESS NAME: DELETION OF STATEMENTS WITH LOW SIGNIFICANCE LEVEL	TYPE: LOCAL	NUMBER OF DEGREE OF PRIORITY: 400
NAME OF COMPUTER PROGRAM: AR_A1	PROCESS NAME: DETECTION OF LOOP OF READ	TYPE: OVERALL	NUMBER OF DEGREE OF PRIORITY: 100
NAME OF COMPUTER PROGRAM: AR_A2	PROCESS NAME: ENTRY OF OVERALL NAME OF VARIABLE	TYPE: OVERALL	NUMBER OF DEGREE OF PRIORITY: 200

FIG.38

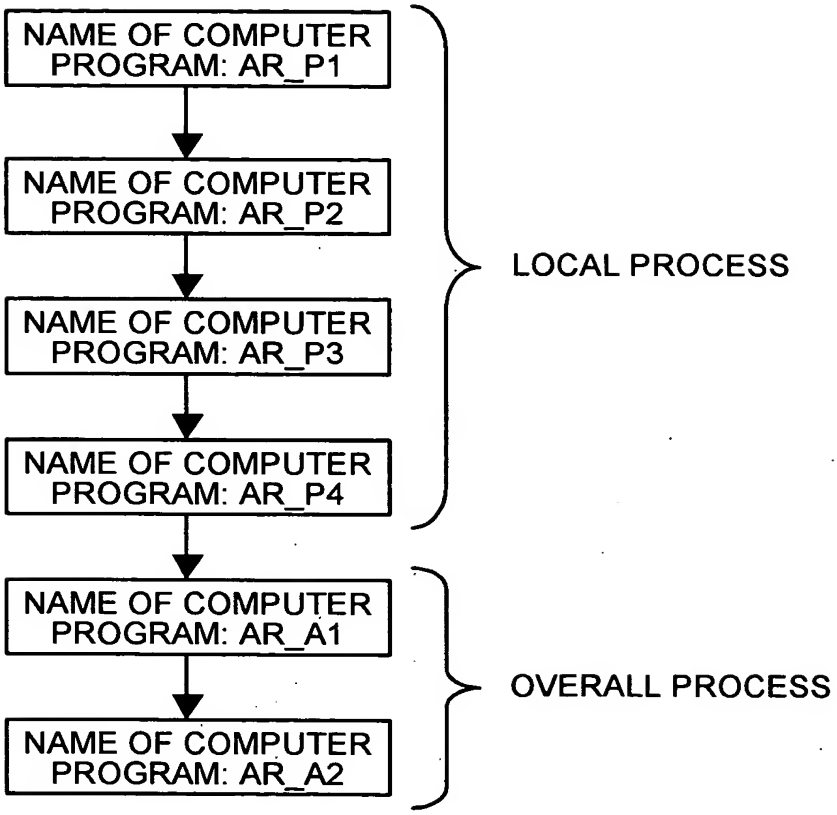

NAME OF COMPUTER PROGRAM: AR_P1
NAME OF COMPUTER PROGRAM: AR_P2
NAME OF COMPUTER PROGRAM: AR_P3
NAME OF COMPUTER PROGRAM: AR_P4
LOCAL PROCESS
NAME OF COMPUTER PROGRAM: AR_A1
NAME OF COMPUTER PROGRAM: AR_A2
OVERALL PROCESS

FIG.39

```
< loop_read_file statement_priority ="1">
    <file name="INFILE">
     <record>< var  name="IN- RECORD" . data_priority ="3"/></record>
    </file>
   <until>< at_end_file  name="INFILE"/></until>
    <sequences qt="2">
     <if  statement_priority ="2">
        <condition>
          <expression>
           <var  name="IN - RECORD.FAMILYDATA"  data_priority ="1"/>
           <comparison_operator  name="="/>
             <constant value="ZERO"/>
          </expression>
         </condition>
        <else>
          <sequences qt="2">
           < perform_internal statement_priority ="1">
              <condition>
                <varying><var  name="I" . data_priority ="1"/></varying>
                <varying_from><constant value="1" type=" int "/></ varying_from >
                   <by><constant value="1" type=" int "/></by>
                 <until>
                   <expression>
                     <var  name="I"  data_priority ="1"/>
                     <comparison_operator  name="&gt ;"/>
                        <constant value="20" type=" int "/>
                   </expression>
                 </until>
               </condition>
               <sequences qt="2">
                <move statement_priority ="1">
                   <ref part="on" >< var  name="IN - RECORD"  data_priority ="3"/></ref>
                  <def>< var name="OUT- RECORD"  data_priority ="2"/></def>
                </move>
                <write  statement_priority ="1">
                     <file name="OUTFILE">
                       <record>< var name="OUT- RECORD"  data_priority ="2"/></record>
                     </file>
                 </write>
              </sequences>
            </ perform_internal >
          </sequences>
        </else>
      </if>
   </sequences>
</ loop_read_file >
```

FIG.40

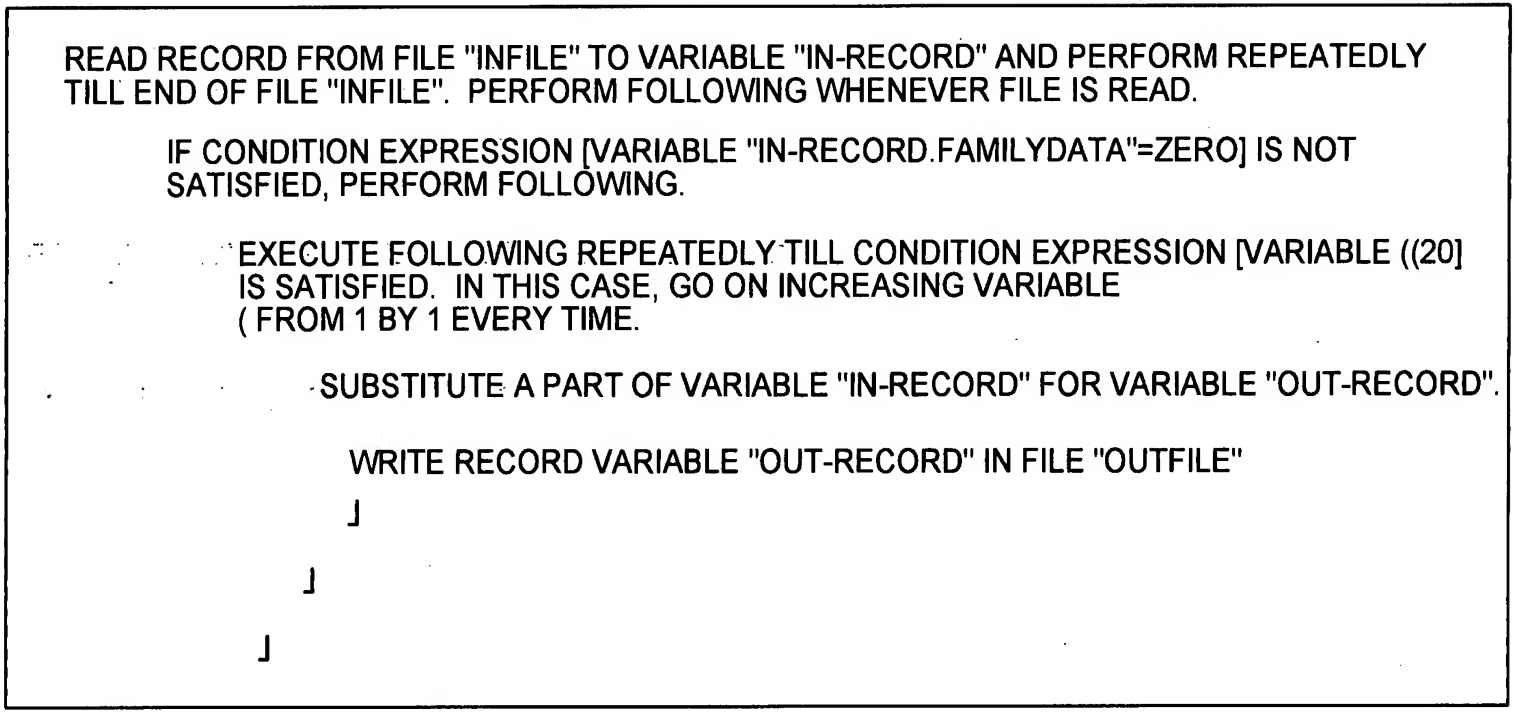

READ RECORD FROM FILE "INFILE" TO VARIABLE "IN-RECORD" AND PERFORM REPEATEDLY TILL END OF FILE "INFILE". PERFORM FOLLOWING WHENEVER FILE IS READ.
IF CONDITION EXPRESSION [VARIABLE "IN-RECORD.FAMILYDATA"=ZERO] IS NOT SATISFIED, PERFORM FOLLOWING.
EXECUTE FOLLOWING REPEATEDLY TILL CONDITION EXPRESSION [VARIABLE ((20] IS SATISFIED. IN THIS CASE, GO ON INCREASING VARIABLE (FROM 1 BY 1 EVERY TIME.
SUBSTITUTE A PART OF VARIABLE "IN-RECORD" FOR VARIABLE "OUT-RECORD".
WRITE RECORD VARIABLE "OUT-RECORD" IN FILE "OUTFILE"

FIG.41

COMPUTER PROGRAM OUTLINE	TIME AND DATE OF PREPARATION
	13:45, 11/07/2003

NAME OF COMPUTER PROGRAM	TESTSAMPLE	FILE NAME	TESTSAMPLE.COB

COMMENT COLUMN

FILE INFORMATION

No.	FILE NAME	EXTERNAL NAME	TYPE
1.	INFILE	INFILE	COBOL FILE
2.	OUTFILE	OUTFILE	COBOL FILE

COMPUTER PROGRAM OUTLINE

READ RECORD FROM FILE "INFILE" TO VARIABLE "IN-RECORD" AND PERFORM REPEATEDLY TILL END OF FILE "INFILE". PERFORM FOLLOWING WHENEVER FILE IS READ.

「IF CONDITION EXPRESSION [VARIABLE "IN-RECORD.FAMILYDATA"=ZERO] IS NOT SATISFIED, PERFORM FOLLOWING.

「EXECUTE FOLLOWING REPEATEDLY TILL CONDITION EXPRESSION [VARIABLE ((20] IS SATISFIED. IN THIS CASE, GO ON INCREASING VARIABLE (FROM 1 BY 1 EVERY TIME.

「SUBSTITUTE A PART OF VARIABLE "IN-RECORD" FOR VARIABLE "OUT-RECORD".

WRITE RECORD VARIABLE "OUT-RECORD" IN FILE "OUTILE".

」

」

」

FIG.42

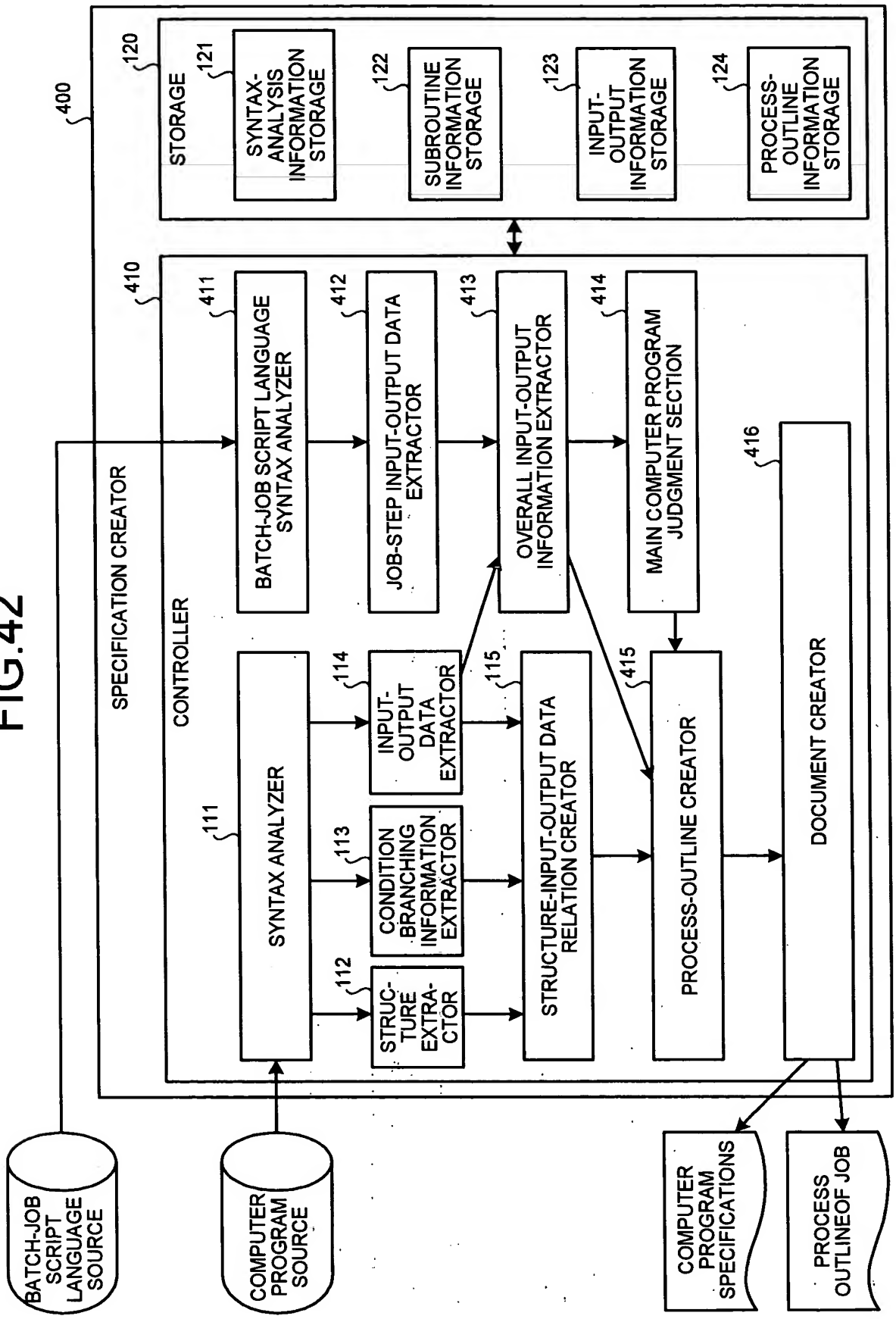

400
SPECIFICATION CREATOR
120
STORAGE
121
SYNTAX-ANALYSIS INFORMATION STORAGE
122
SUBROUTINE INFORMATION STORAGE
123
INPUT-OUTPUT INFORMATION STORAGE
124
PROCESS-OUTLINE INFORMATION STORAGE
410
CONTROLLER
411
BATCH-JOB SCRIPT LANGUAGE SYNTAX ANALYZER
412
JOB-STEP INPUT-OUTPUT DATA EXTRACTOR
413
OVERALL INPUT-OUTPUT INFORMATION EXTRACTOR
414
MAIN COMPUTER PROGRAM JUDGMENT SECTION
416
DOCUMENT CREATOR
111
SYNTAX ANALYZER
114
INPUT-OUTPUT DATA EXTRACTOR
115
STRUCTURE-INPUT-OUTPUT DATA RELATION CREATOR
415
PROCESS-OUTLINE CREATOR
113
CONDITION BRANCHING INFORMATION EXTRACTOR
112
STRUC-TURE EXTRA-CTOR
BATCH-JOB SCRIPT LANGUAGE SOURCE
COMPUTER PROGRAM SOURCE
COMPUTER PROGRAM SPECIFICATIONS
PROCESS OUTLINEOF JOB

FIG.43

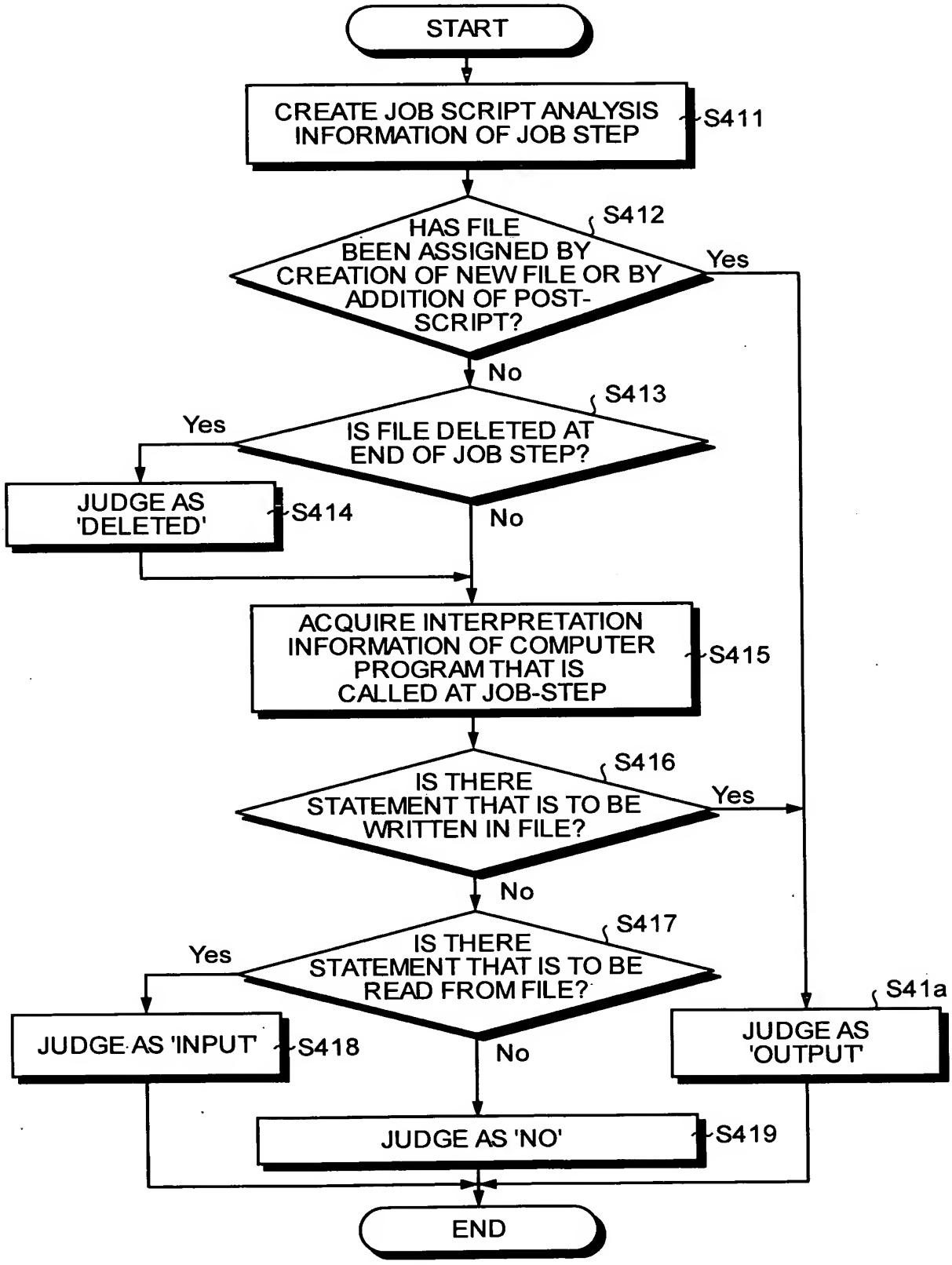

START
CREATE JOB SCRIPT ANALYSIS INFORMATION OF JOB STEP
S411
HAS FILE BEEN ASSIGNED BY CREATION OF NEW FILE OR BY ADDITION OF POST-SCRIPT?
S412
Yes
No
IS FILE DELETED AT END OF JOB STEP?
S413
Yes
No
JUDGE AS 'DELETED'
S414
ACQUIRE INTERPRETATION INFORMATION OF COMPUTER PROGRAM THAT IS CALLED AT JOB-STEP
S415
IS THERE STATEMENT THAT IS TO BE WRITTEN IN FILE?
S416
Yes
No
IS THERE STATEMENT THAT IS TO BE READ FROM FILE?
S417
Yes
No
JUDGE AS 'INPUT'
S418
JUDGE AS 'OUTPUT'
S41a
JUDGE AS 'NO'
S419
END

FIG.44

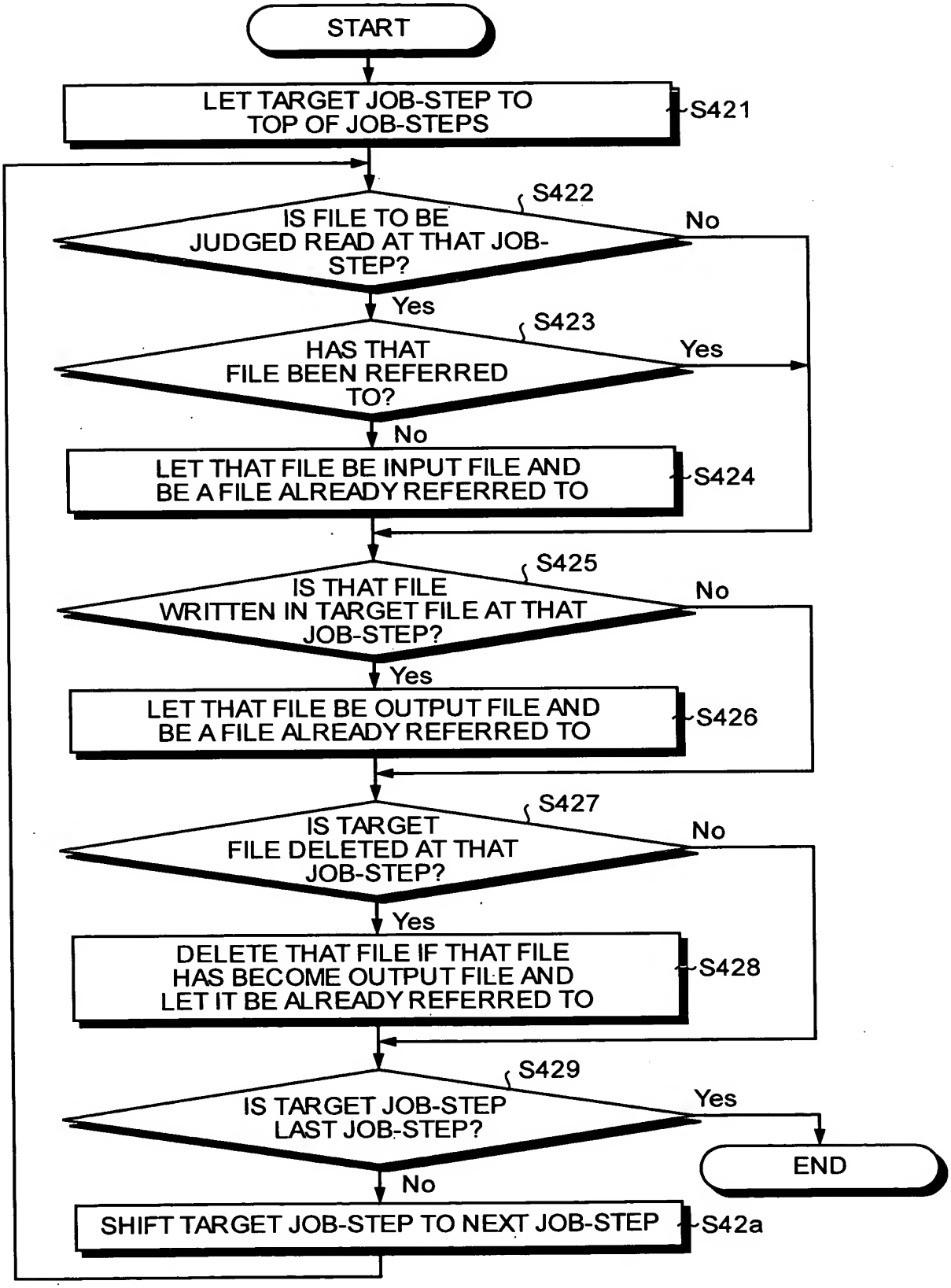

START
LET TARGET JOB-STEP TO TOP OF JOB-STEPS
S421
S422
IS FILE TO BE JUDGED READ AT THAT JOB-STEP?
No
Yes
S423
HAS THAT FILE BEEN REFERRED TO?
Yes
No
LET THAT FILE BE INPUT FILE AND BE A FILE ALREADY REFERRED TO
S424
S425
IS THAT FILE WRITTEN IN TARGET FILE AT THAT JOB-STEP?
No
Yes
LET THAT FILE BE OUTPUT FILE AND BE A FILE ALREADY REFERRED TO
S426
S427
IS TARGET FILE DELETED AT THAT JOB-STEP?
No
Yes
DELETE THAT FILE IF THAT FILE HAS BECOME OUTPUT FILE AND LET IT BE ALREADY REFERRED TO
S428
S429
IS TARGET JOB-STEP LAST JOB-STEP?
Yes
END
No
SHIFT TARGET JOB-STEP TO NEXT JOB-STEP
S42a

FIG.45

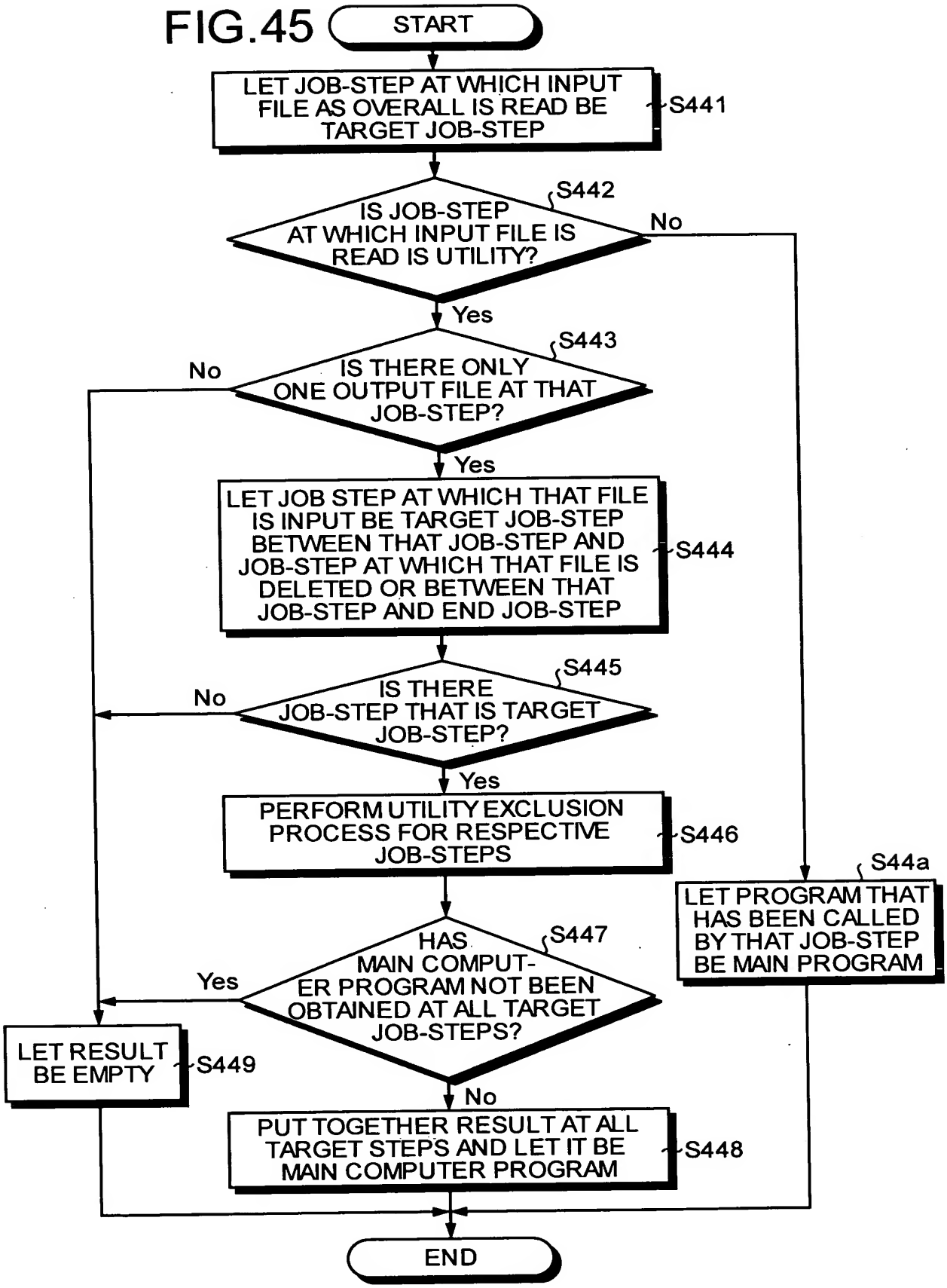

START
LET JOB-STEP AT WHICH INPUT FILE AS OVERALL IS READ BE TARGET JOB-STEP
S441
S442
IS JOB-STEP AT WHICH INPUT FILE IS READ IS UTILITY?
No
Yes
S443
IS THERE ONLY ONE OUTPUT FILE AT THAT JOB-STEP?
No
Yes
LET JOB STEP AT WHICH THAT FILE IS INPUT BE TARGET JOB-STEP BETWEEN THAT JOB-STEP AND JOB-STEP AT WHICH THAT FILE IS DELETED OR BETWEEN THAT JOB-STEP AND END JOB-STEP
S444
S445
IS THERE JOB-STEP THAT IS TARGET JOB-STEP?
No
Yes
PERFORM UTILITY EXCLUSION PROCESS FOR RESPECTIVE JOB-STEPS
S446
S44a
LET PROGRAM THAT HAS BEEN CALLED BY THAT JOB-STEP BE MAIN PROGRAM
S447
HAS MAIN COMPUTER PROGRAM NOT BEEN OBTAINED AT ALL TARGET JOB-STEPS?
Yes
No
LET RESULT BE EMPTY
S449
PUT TOGETHER RESULT AT ALL TARGET STEPS AND LET IT BE MAIN COMPUTER PROGRAM
S448
END

FIG.46

```
///JOB01 JOB
//*****************************************
//STEP1 EXEC PGM=PROGA
//IN01  DD DSN=DENPYO01,DISP=SHR
//OT01  DD DSN=WORK1,DISP=(NEW,PASS)
//*****************************************
//STEP2 EXEC PGM=SORT
//SORTIN  DD DSN=WORK1,DISP=(OLD,DELETE)
//SORTOT  DD DSN=WORK2,DISP=(NEW,PASS)
//*****************************************
//STEP3 EXEC PGM=PROGB
//IN01  DD DSN=WORK2,DISP=(OLD,DELETE)
//OT01  DD DSN=SYUKEI1,DISP(NEW,PASS)
//*****************************************
//STEP4 EXEC PGM=ENDMSG
```

FIG.47

STEP NAME	NAME OF PROGRAM CALLED	FILE							
		DEPYO1		WORK1		WORK2		DATA1	
		EXTERNAL NAME	MODE	EXTERNAL NAME	MODE	EXTERNAL NAME	MODE	EXTERNAL NAME	MODE
STEP1	PROGA	IN01	S	OT01	NP				
STEP2	SORT			SOTRIN	OD	SORTOT	NP		
STEP3	PROGB					IN01	OD	OT01	NP
STEP4	ENDMSG								

FIG.48

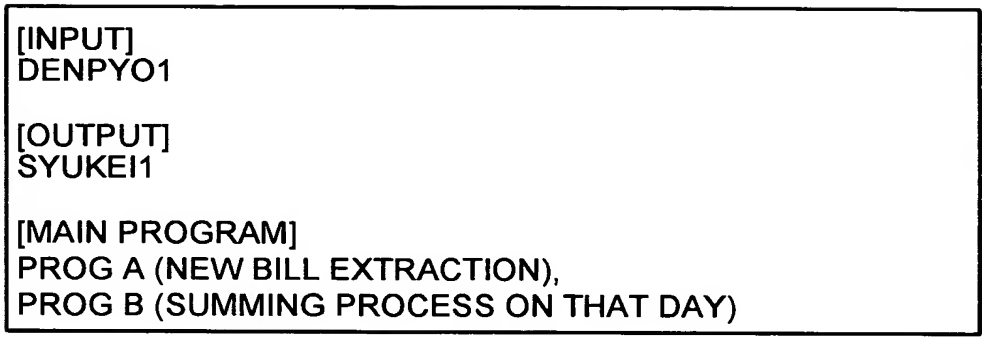

[INPUT]
DENPYO1
[OUTPUT]
SYUKEI1
[MAIN PROGRAM]
PROG A (NEW BILL EXTRACTION),
PROG B (SUMMING PROCESS ON THAT DAY)

FIG.49

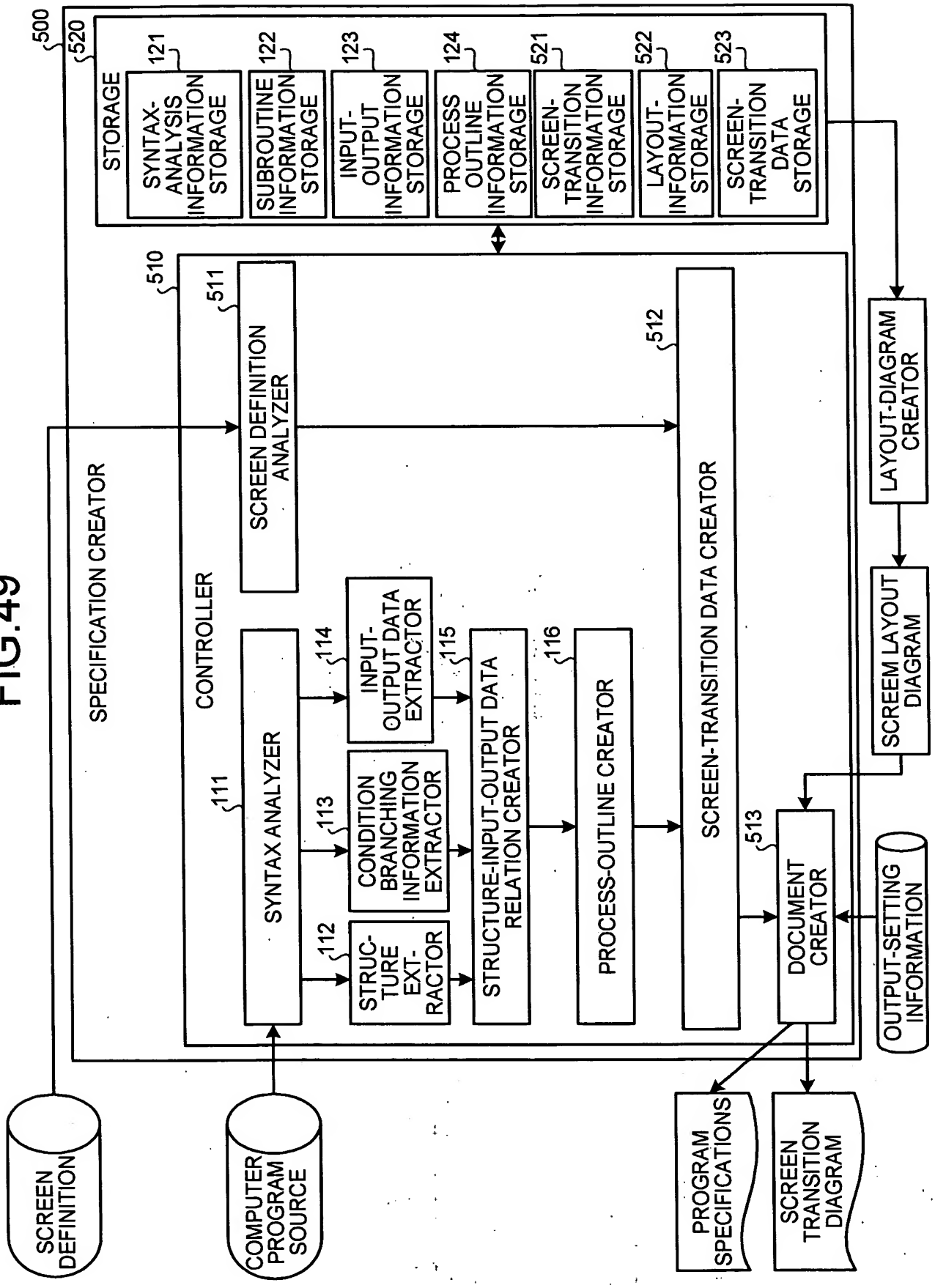

SPECIFICATION CREATOR
500
520
STORAGE
121
SYNTAX-ANALYSIS INFORMATION STORAGE
122
SUBROUTINE INFORMATION STORAGE
123
INPUT-OUTPUT INFORMATION STORAGE
124
PROCESS OUTLINE INFORMATION STORAGE
521
SCREEN-TRANSITION INFORMATION STORAGE
522
LAYOUT-INFORMATION STORAGE
523
SCREEN-TRANSITION DATA STORAGE
510
CONTROLLER
511
SCREEN DEFINITION ANALYZER
111
SYNTAX ANALYZER
112
STRUC-TURE EXT-RACTOR
113
CONDITION BRANCHING INFORMATION EXTRACTOR
114
INPUT-OUTPUT DATA EXTRACTOR
115
STRUCTURE-INPUT-OUTPUT DATA RELATION CREATOR
116
PROCESS-OUTLINE CREATOR
512
SCREEN-TRANSITION DATA CREATOR
513
DOCUMENT CREATOR
LAYOUT-DIAGRAM CREATOR
SCREEM LAYOUT DIAGRAM
OUTPUT-SETTING INFORMATION
SCREEN DEFINITION
COMPUTER PROGRAM SOURCE
PROGRAM SPECIFICATIONS
SCREEN TRANSITION DIAGRAM

FIG.50

```
DFGMA01  FID     TYPE=DSP PGM=STD
         DEVICE
         FRAME
          PART    CONTROL=RETRY
      '&'     SCHAR   TYPE=ATTR,FIELD=IN,ATTR=(IDH,ALP,OTL        ,MDT,CUR)
         LAYOUT
  #


         LAYEND
    DXCGUA01  RECORD   TYPE=INOUT,NAME='  MGA01 - MENU SCREEN RECORD'
    AID      DATA    USAGE=AID,NAME='   MGA01 - ATTENTION '
    H        DATA    USAGE=CTRL3,PIC=(X,1),NAME='  MGA01 - JUDGMENT'
    H        DIST    DATA=('Y',ELSE)
                 PARM=('F.M05000','F.DFGMA01')
          END
```

LAYOUT INFORMATION

SCREEN-TRANSITION INFORMATION

FIG.51

TRANSITION-ORIGIN SCREEN	TRANSITION CONDITION	TRANSITION-TARGET SCREEN	CLASSIFICATION OF TRANSITION TARGET
M05000	SELECTION PROCESS = 1	M05010	SCREEN
M05000	SELECTION PROCESS = 2	PG0001	COMPUTER PROGRAM

FIG.52

```
<Descripiton   id="00000001" name="M05000" type="display" source="M05000.psm"    layer=""  layermax =""><
    <outline>   SERVICE   </outline>
    <condition expression="SERECTION PROCESS=1">
        <Description id="00000002" name="M05010" type="display"    source="M05010.psm">
            <outline> MOD  PARAMETER </outline>
                ...
        <Description/>
    </condition>
    <condition expression=" SERECTION PROCESS =2">
        <Description id="00000003" name="M05020" type="display"    filename="M05020.psm">
            <outline> XBOST PARAMETER </outline>
                ...
        </Description>
    </condition>
</ Descripiton >
```

FIG.53A

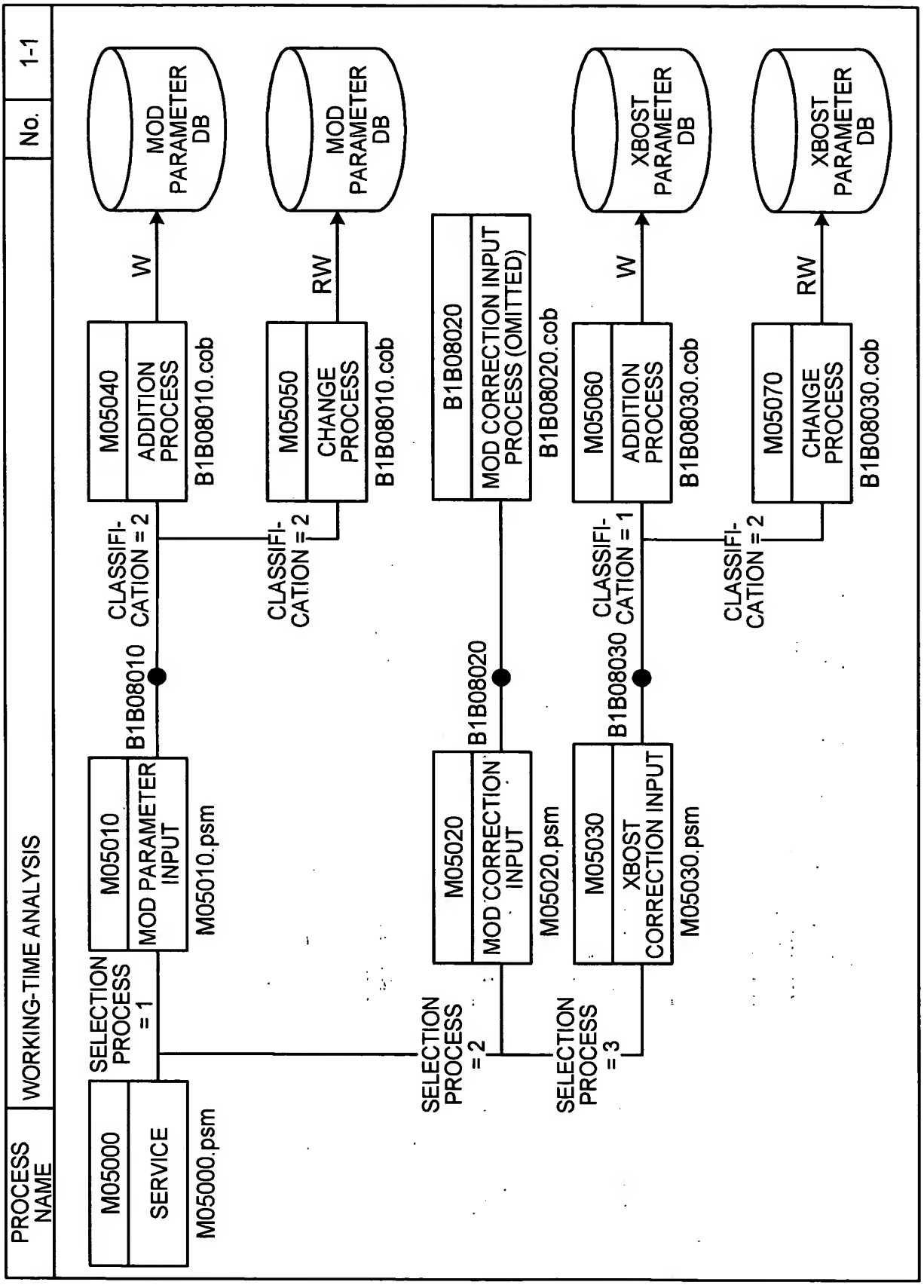

PROCESS NAME
WORKING-TIME ANALYSIS
No.
1-1
M05000
SERVICE
M05000.psm
SELECTION PROCESS = 1
M05010
MOD PARAMETER INPUT
M05010.psm
B1B08010
CLASSIFI-CATION = 2
M05040
ADDITION PROCESS
B1B08010.cob
W
MOD PARAMETER DB
CLASSIFI-CATION = 2
M05050
CHANGE PROCESS
B1B08010.cob
RW
MOD PARAMETER DB
SELECTION PROCESS = 2
M05020
MOD CORRECTION INPUT
M05020.psm
B1B08020
B1B08020
MOD CORRECTION INPUT PROCESS (OMITTED)
B1B08020.cob
SELECTION PROCESS = 3
M05030
XBOST CORRECTION INPUT
M05030.psm
B1B08030
CLASSIFI-CATION = 1
M05060
ADDITION PROCESS
B1B08030.cob
W
XBOST PARAMETER DB
CLASSIFI-CATION = 2
M05070
CHANGE PROCESS
B1B08030.cob
RW
XBOST PARAMETER DB

FIG.53B

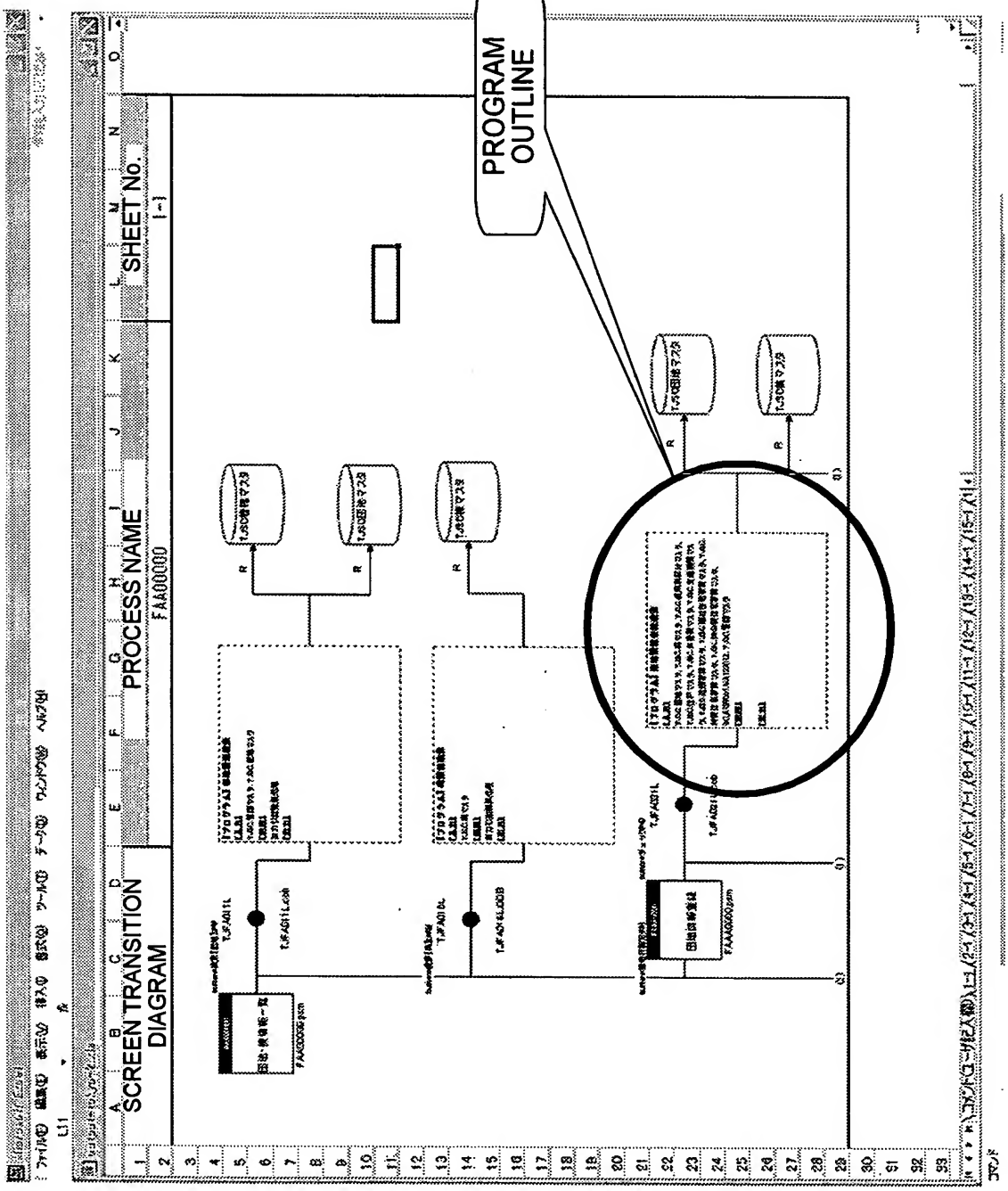

SCREEN TRANSITION DIAGRAM
PROCESS NAME
FAA00000
SHEET No.
PROGRAM OUTLINE

FIG.54

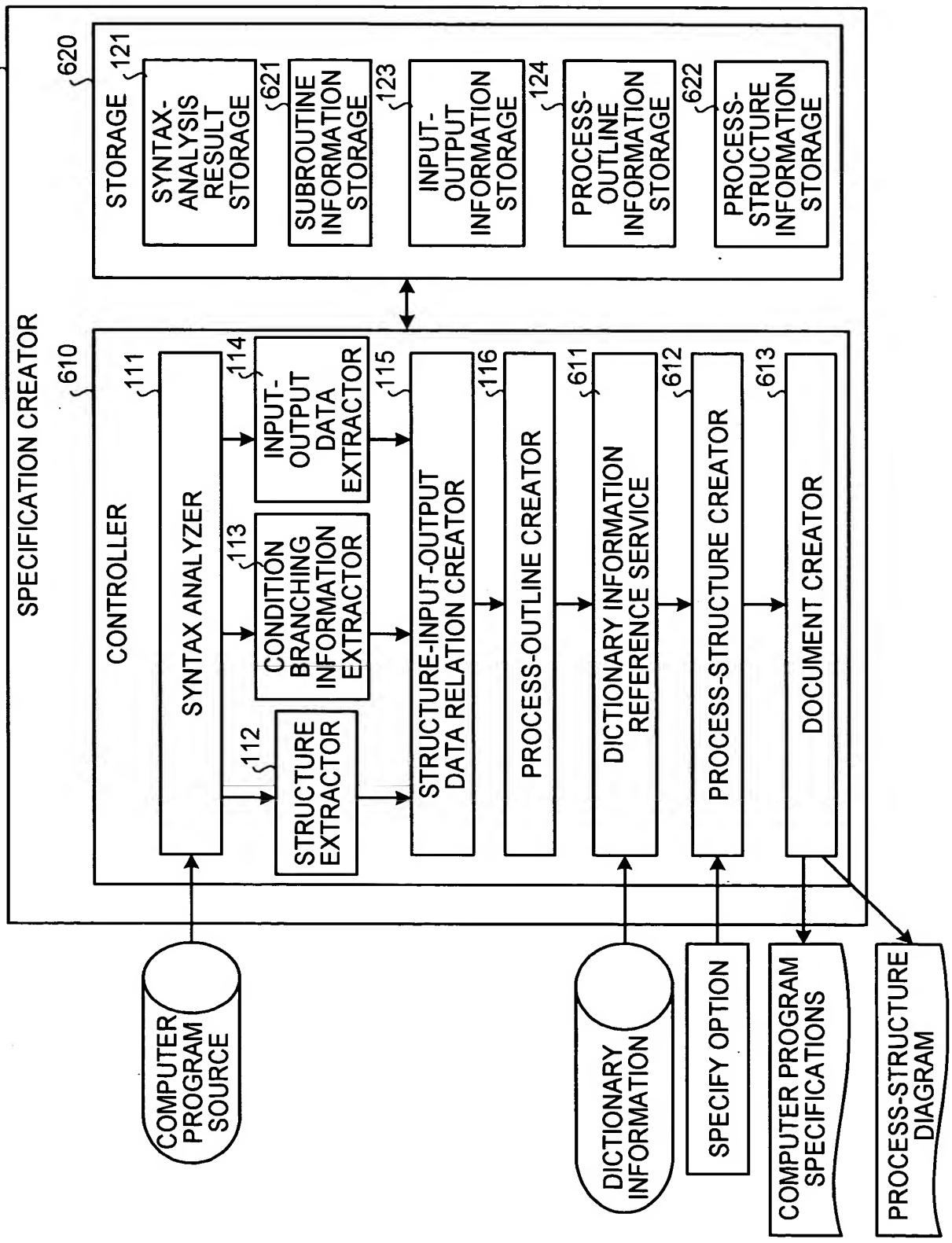

600
SPECIFICATION CREATOR
620
STORAGE
121
SYNTAX-ANALYSIS RESULT STORAGE
621
SUBROUTINE INFORMATION STORAGE
123
INPUT-OUTPUT INFORMATION STORAGE
124
PROCESS-OUTLINE INFORMATION STORAGE
622
PROCESS-STRUCTURE INFORMATION STORAGE
610
CONTROLLER
111
SYNTAX ANALYZER
112
STRUCTURE EXTRACTOR
113
CONDITION BRANCHING INFORMATION EXTRACTOR
114
INPUT-OUTPUT DATA EXTRACTOR
115
STRUCTURE-INPUT-OUTPUT DATA RELATION CREATOR
116
PROCESS-OUTLINE CREATOR
611
DICTIONARY INFORMATION REFERENCE SERVICE
612
PROCESS-STRUCTURE CREATOR
613
DOCUMENT CREATOR
COMPUTER PROGRAM SOURCE
DICTIONARY INFORMATION
SPECIFY OPTION
COMPUTER PROGRAM SPECIFICATIONS
PROCESS-STRUCTURE DIAGRAM

FIG.55

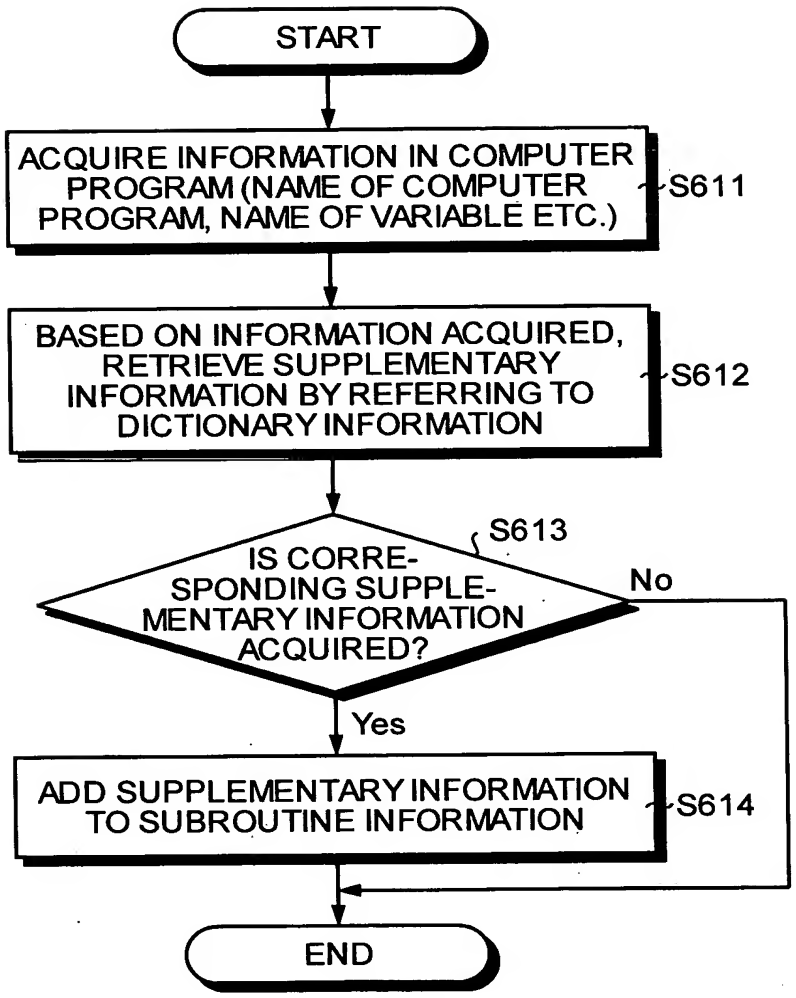

START
ACQUIRE INFORMATION IN COMPUTER PROGRAM (NAME OF COMPUTER PROGRAM, NAME OF VARIABLE ETC.)
S611
BASED ON INFORMATION ACQUIRED, RETRIEVE SUPPLEMENTARY INFORMATION BY REFERRING TO DICTIONARY INFORMATION
S612
S613
IS CORRE-SPONDING SUPPLE-MENTARY INFORMATION ACQUIRED?
No
Yes
ADD SUPPLEMENTARY INFORMATION TO SUBROUTINE INFORMATION
S614
END

FIG.56

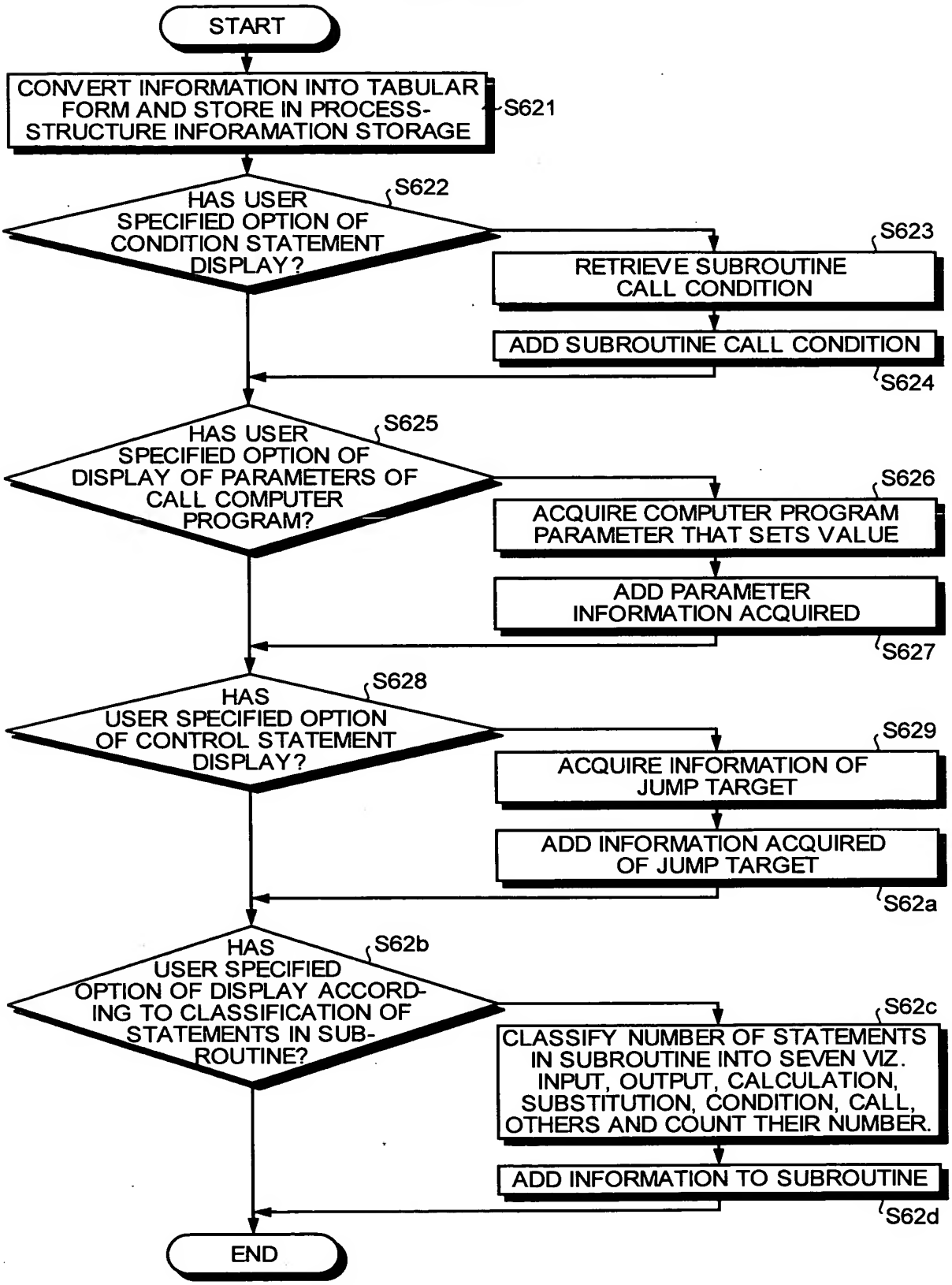

START
CONVERT INFORMATION INTO TABULAR FORM AND STORE IN PROCESS-STRUCTURE INFORAMATION STORAGE
S621
HAS USER SPECIFIED OPTION OF CONDITION STATEMENT DISPLAY?
S622
RETRIEVE SUBROUTINE CALL CONDITION
S623
ADD SUBROUTINE CALL CONDITION
S624
HAS USER SPECIFIED OPTION OF DISPLAY OF PARAMETERS OF CALL COMPUTER PROGRAM?
S625
ACQUIRE COMPUTER PROGRAM PARAMETER THAT SETS VALUE
S626
ADD PARAMETER INFORMATION ACQUIRED
S627
HAS USER SPECIFIED OPTION OF CONTROL STATEMENT DISPLAY?
S628
ACQUIRE INFORMATION OF JUMP TARGET
S629
ADD INFORMATION ACQUIRED OF JUMP TARGET
S62a
HAS USER SPECIFIED OPTION OF DISPLAY ACCORDING TO CLASSIFICATION OF STATEMENTS IN SUB-ROUTINE?
S62b
CLASSIFY NUMBER OF STATEMENTS IN SUBROUTINE INTO SEVEN VIZ. INPUT, OUTPUT, CALCULATION, SUBSTITUTION, CONDITION, CALL, OTHERS AND COUNT THEIR NUMBER.
S62c
ADD INFORMATION TO SUBROUTINE
S62d
END

FIG.57A

```
PROGRAM-ID PROGA
   ************ PROCEDURE DIVISION ****************************
PROCEDURE DIVISION.

      OPEN   INPUT   INFILE.
      DISPLAY "## PROGRAM START " UPON CONSOLE.

      PERFORM   READ-SECT.
      IF END-FLAG  = "END"
           DISPLAY "## NOT EVEN ONE RECORD CAN BE READ." UPON CONSOLE
           DISPLAY "## ERROR IS ENDED. " UPON CONSOLE
           STOP RUN
       END-IF.

      PERFORM   MAIN-SECT  UNTIL   END-FLAG = "END"

      CLOSE   INFILE.
      DISPLAY "NUMBER OF OUTPUTS    =[" OUT-COUNT  "]" UPON CONSOLE.
      DISPLAY "ENDED NORMALLY.  "  UPON CONSOLE.

     STOP RUN.
   ************ READ ROUTINE ************************************
READ-SECT    SECTION.
       READ   INFILE
            AT END       MOVE  "END" TO   END-FLAG
            NOT AT END   ADD   1   TO   IN-COUNT
       END-READ.
READ-SECT-END.
       EXIT.
   ************ MAIN ROUTINE ************************************
MAIN-SECT   SECTION.
       IF FAMILYDATA = ZERO
                 ADD    1      TO     SKIP-COUNT
      ELSE
                 ADD    1      TO     PROC-COUNT
            PERFORM VARYING I FROM 1 BY 1
                                    UNTIL I > 20 OR F-MEMBER(I) = ZERO
                  MOVE   MEMBERCODE   TO  PARA1-MEMBERCODE
                  CALL      PROGW      USING PARA1
                       ADD    1          TO  OUT-COUNT
            END-PERFORM
       END-IF.

      PERFORM   READ-SECT.
MAIN-SECT-END.
      EXIT.
```

FIG.57B

```
PROGRAM-ID PROGW
 ************ PROCEDURE DIVISION ****************************
PROCEDURE DIVISION USING PARA1.

    OPEN   OUTPUT  OUTFILE.
    MOVE   PARA1       TO  OUT-RECORD.
    WRITE  OUT-RECORD
    CLOSE  OUTFILE.

    STOP RUN.


 ************ READ ROUTINE ************************************
```

FIG.58

DOCUMENT NAME	PROCESS-STRUCTURE DIAGRAM

NAME OF COMPUTER PROGRAM	SECTION (FIRST NESTING LEVEL)	SECTION (SECOND NESTING LEVEL)	SECTION (THIRD NESTING LEVEL)	READ FILE	WRITE FILE
PROGA					
	first section (no name)				
		READ-SECT		INF(INFILE)	
		MAIN-SECT			
			CALL ""PROGW""		OTF(OUTFILE)

FIG.59

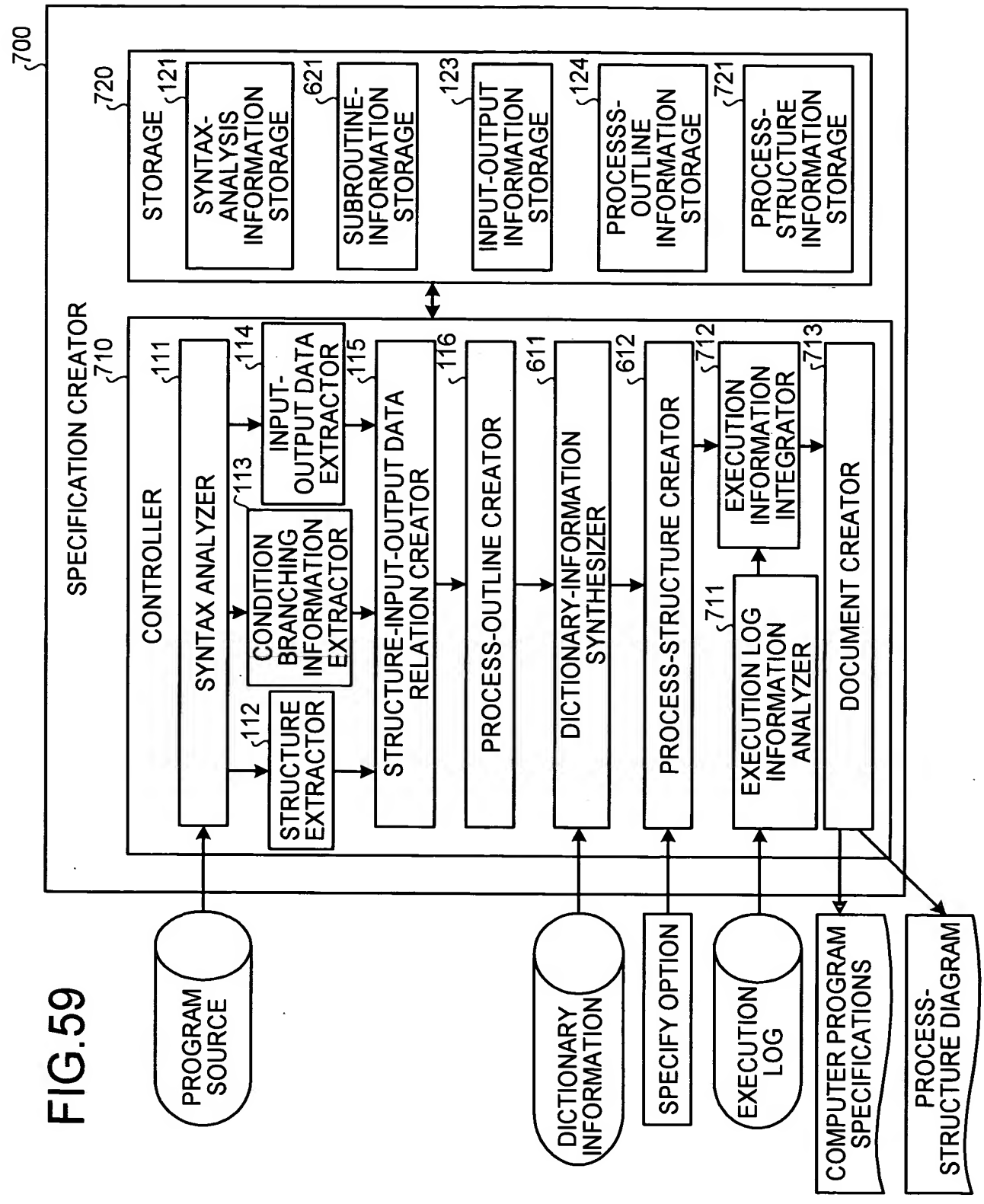

700
SPECIFICATION CREATOR
710
CONTROLLER
111
SYNTAX ANALYZER
112
STRUCTURE EXTRACTOR
113
CONDITION BRANCHING INFORMATION EXTRACTOR
114
INPUT-OUTPUT DATA EXTRACTOR
115
STRUCTURE-INPUT-OUTPUT DATA RELATION CREATOR
116
PROCESS-OUTLINE CREATOR
611
DICTIONARY-INFORMATION SYNTHESIZER
612
PROCESS-STRUCTURE CREATOR
712
EXECUTION INFORMATION INTEGRATOR
711
EXECUTION LOG INFORMATION ANALYZER
713
DOCUMENT CREATOR
720
STORAGE
121
SYNTAX-ANALYSIS INFORMATION STORAGE
621
SUBROUTINE-INFORMATION STORAGE
123
INPUT-OUTPUT INFORMATION STORAGE
124
PROCESSS-OUTLINE INFORMATION STORAGE
721
PROCESS-STRUCTURE INFORMATION STORAGE
PROGRAM SOURCE
DICTIONARY INFORMATION
SPECIFY OPTION
EXECUTION LOG
COMPUTER PROGRAM SPECIFICATIONS
PROCESS-STRUCTURE DIAGRAM

FIG.60

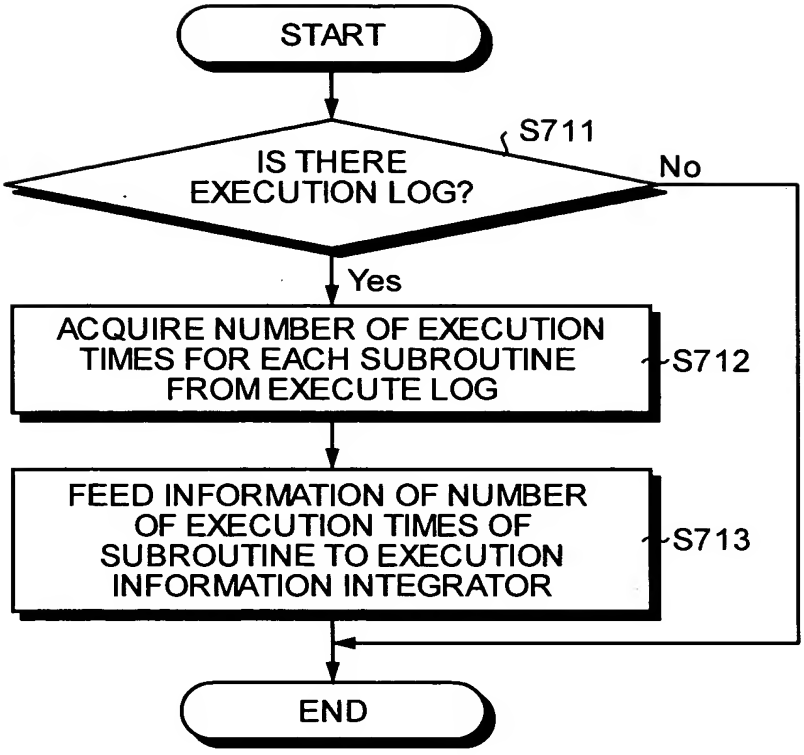

START
S711
IS THERE EXECUTION LOG?
No
Yes
ACQUIRE NUMBER OF EXECUTION TIMES FOR EACH SUBROUTINE FROM EXECUTE LOG
S712
FEED INFORMATION OF NUMBER OF EXECUTION TIMES OF SUBROUTINE TO EXECUTION INFORMATION INTEGRATOR
S713
END

FIG.61

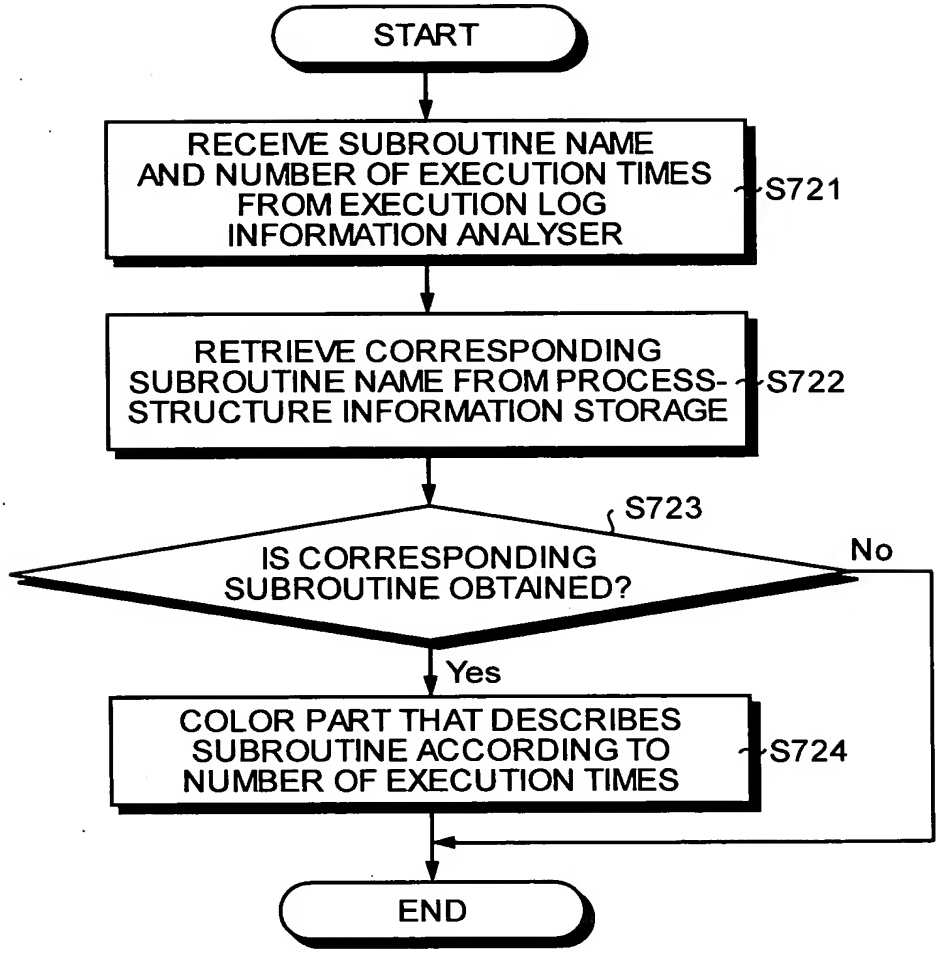

START
RECEIVE SUBROUTINE NAME AND NUMBER OF EXECUTION TIMES FROM EXECUTION LOG INFORMATION ANALYSER
S721
RETRIEVE CORRESPONDING SUBROUTINE NAME FROM PROCESS-STRUCTURE INFORMATION STORAGE
S722
S723
IS CORRESPONDING SUBROUTINE OBTAINED?
No
Yes
COLOR PART THAT DESCRIBES SUBROUTINE ACCORDING TO NUMBER OF EXECUTION TIMES
S724
END

FIG.62

DOCUMENT NAME	PROCESS-STRUCTURE DIAGRAM

NAME OF PROGRAM	SECTION (FIRST NESTING LEVEL)	SECTION (SECOND NESTING LEVEL)	SECTION (THIRD NESTING LEVEL)	READ FILE	WRITE FILE
PROGA					
	first section (no name)				
	NUMBER OF STATEMENTS ACCORDING TO CLASSIFICATION [INPUT: 1, OUTPUT: 5, CALL: 2, CONDITION: 1, OTHERS: 9]	READ-SECT		INF(INFILE)	
		NUMBER OF STATEMENTS [INPUT: 1, CALCULATION: 1, SUBSTITUTION: 1]	<READ-SECT-END>		
		MAIN-SECT			
		EXECUTION CONDITION UNTIL(END-FLAG = "END")	CALL ""PROGW""		OTF(OUTFILE)
		NUMBER OF STATEMENTS [CALCULATION: 3, CALL: 3, SUBSTITUTION: 1, CONDITION: 1]	USING [PARA1-MEMBERCODE=MEMBERCODE]		
			<MAIN-SECT-END>		

EXECUTION INFORMATION: ONCE, TWICE

FIG.63

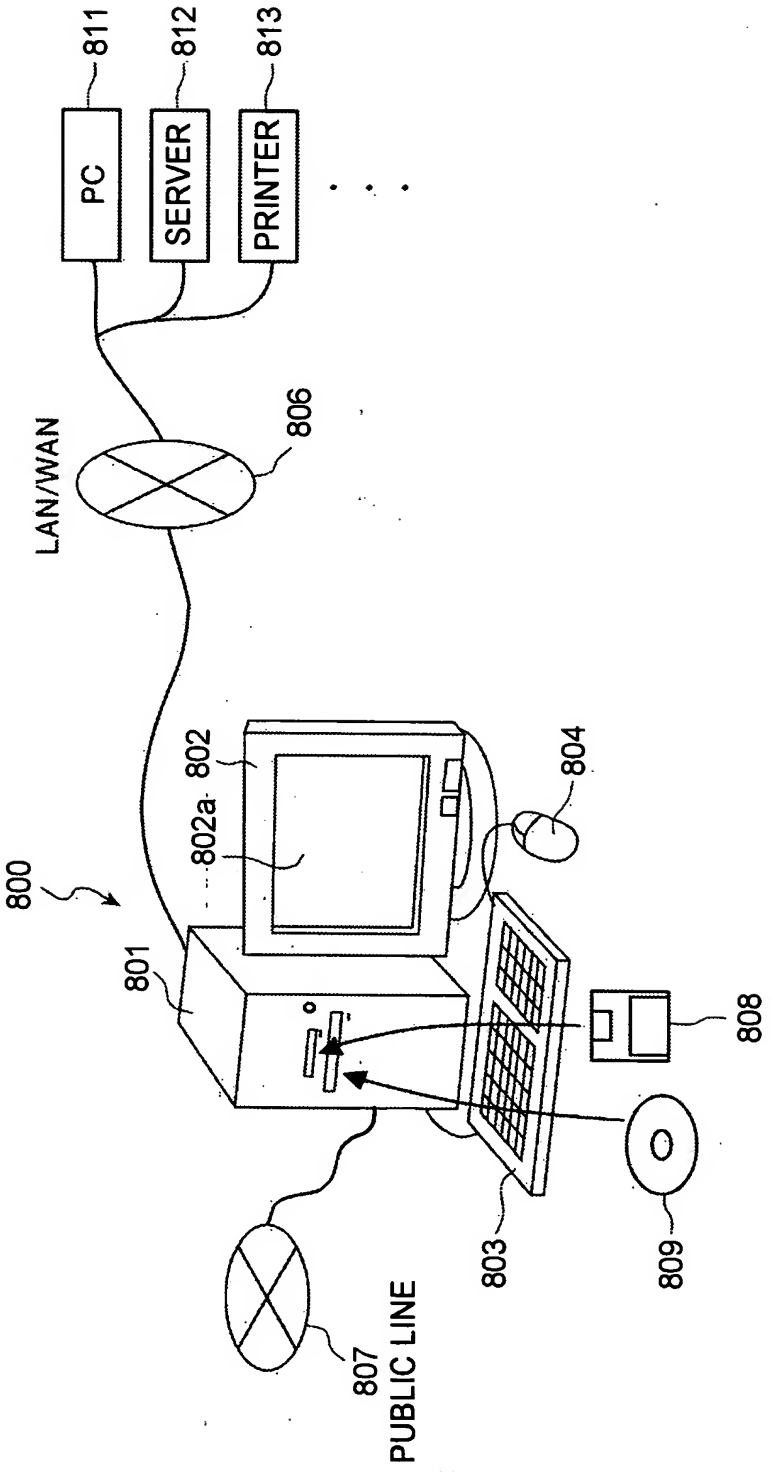

811
812
813
PC
SERVER
PRINTER
LAN/WAN
806
800
802
802a
804
801
808
803
809
807
PUBLIC LINE

FIG.64

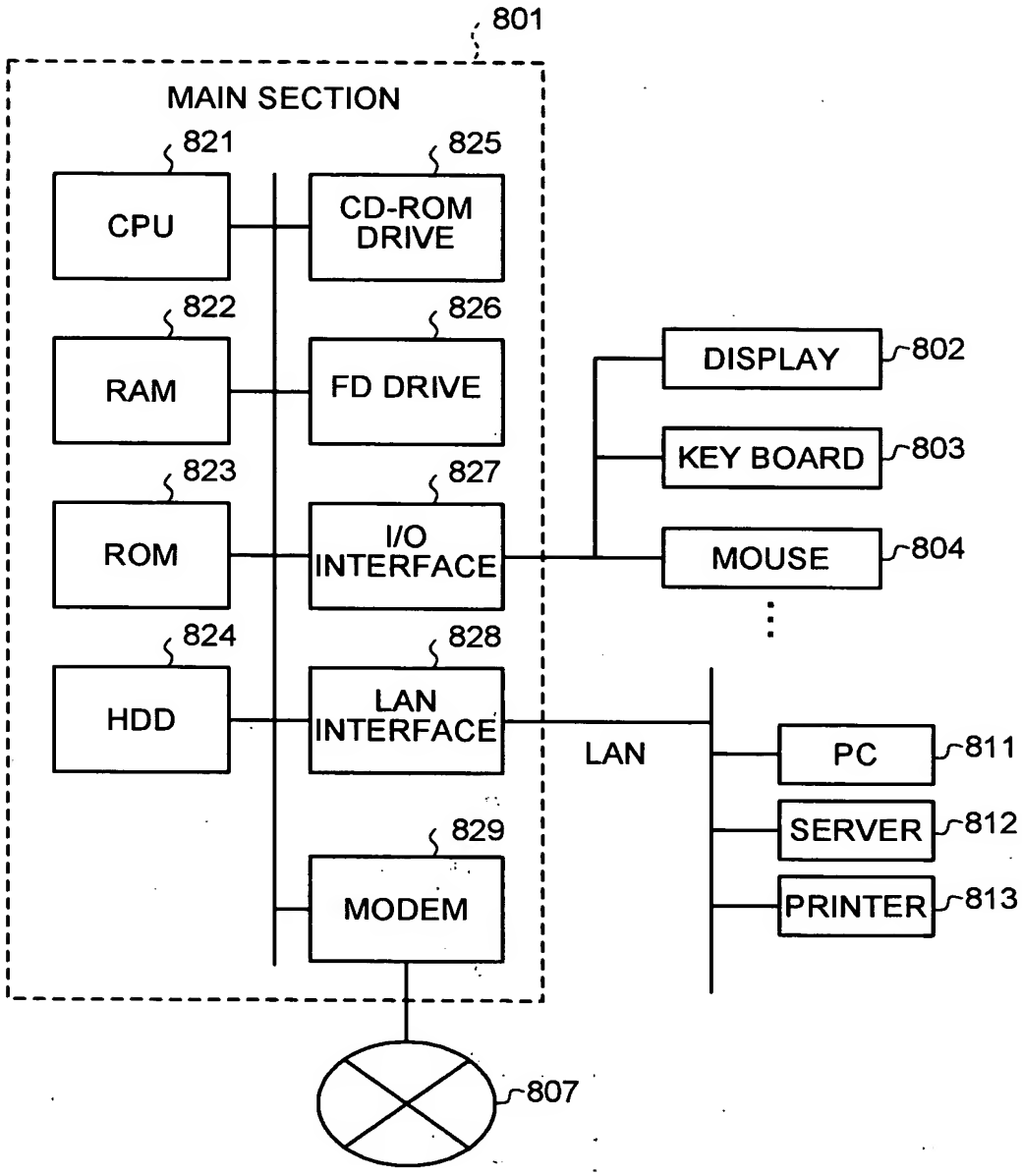

801
MAIN SECTION
821
CPU
825
CD-ROM DRIVE
822
RAM
826
FD DRIVE
823
ROM
827
I/O INTERFACE
824
HDD
828
LAN INTERFACE
829
MODEM
807
DISPLAY
802
KEY BOARD
803
MOUSE
804
LAN
PC
811
SERVER
812
PRINTER
813

FIG.65